



Nortel IVR 2.5/S

SNA Host Communications User Guide

Product release 2.5/S

Standard 1.0

July 1997

NORTEL
NORTHERN TELECOM

Nortel IVR 2.5/S

SNA Host Communications User Guide

Product release: 2.5/S
Document release: Standard 1.0
Date: July 1997

© 1997 Northern Telecom
All rights reserved

Printed in the United States of America

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

IVR and Nortel are registered trademarks of Northern Telecom in the U.S. and Canada. IVR Generator 2.0/S and Meridian 1 are registered trademarks of Northern Telecom. DEC and VT 420 are registered trademarks of Digital Equipment Corporation. HP, LaserJet, and ThinkJet are registered trademarks of Hewlett-Packard Company. X Window System and X are trademarks of the Massachusetts Institute of Technology. Motif is a trademark of Open Software Foundation Inc. UNIX is a registered trademark of AT&T.

Publication history

July 1997

Standard 1.0 release of the *SNA Host Communications User Guide* for Nortel IVR 2.5/S.

Contents

About this guide ix

Who should use this guide	ix
How to use this guide	ix
Additional manuals	x
Conventions used in this guide	x

1 About host communication links 1-1

The SNA Gateway hardware and software	1-2
Creating IVR Generator applications that communicate with an SNA host	1-4
IVR Generator COM cells.	1-4
TRS	1-6
Action and screen templates	1-6
TRSC	1-6
EXPRESS communications software and hardware	1-6
EHLLAPI.	1-7
Terminal emulation	1-7

2 Installing and configuring the host link software and hardware 2-1

Site planning.	2-2
Pre-site planning	2-2
Package contents.	2-2
Installing the communications hardware	2-3
Installing the EXPRESS software.	2-3
The EXPRESS Server and Desktop	2-4
Installing the EXPRESS SNA Server and Desktop	2-5
EXPRESS Server installation procedure	2-6
Files affected by EXPRESS installation.	2-7
Starting EXPRESS.	2-7

Configuring EXPRESS	2-8
Defining hardware	2-8
Defining communications	2-8
Defining LU 6.2 (5250 only)	2-10
Defining users and sessions	2-12
Setting limits for terminal emulation	2-12
Verifying connections to the host computer	2-13
Checking the status of your links and stations	2-13
Bringing up an interactive session with the host	2-14
Configuring IVR Generator host communications	2-14
Setting up the trs.conf file	2-14
Setting up the trs.sdf file	2-16
Specifying separate IDs/passwords for each LU	2-18
Using the start-up script	2-19
Editing system files (for SCO only)	2-22
Bringing up IVR Generator including TRS	2-22
Start-up switches	2-22
Testing communication with IVR Generator	2-23

3	Creating template files	3-1
	Determining the required transactions	3-1
	Overview of action templates	3-5
	Creating action templates	3-6
	Action Template syntax	3-11
	Initial-Action templates	3-14
	Action templates	3-16
	Reset-Action templates	3-17
	Sample Reset-Action template	3-18
	Logout-Action template	3-19
	Sample Logout-Action template	3-19
	Heartbeat Action template	3-21
	Screen templates	3-22
	Screen template syntax	3-27
	Sample screen template	3-34
	Testing your action and screen templates	3-35
	Using trs -c	3-35
	TRS log	3-38

4	Using the COM cells to access the host computer	4-1
	Setting the COMI cell parameters	4-2
	COMI cell parameters page	4-2
	Setting the COMO cell parameters	4-5
	The COMO cell parameters page	4-5
	Using the COMA cell	4-8
	COMA cell parameters window branches	4-8
	Determining successful COM cell communication	4-9
	A sample application using the COM cells	4-10

5	Troubleshooting	5-1
	Debugging the EXPRESS application	5-1
	Log file	5-1
	Trace file.	5-2
	Background process debugging.	5-2
	Host communication messages	5-5

	Appendix A: List of terms	A-1
--	----------------------------------	------------

	Index	Index-1
--	--------------	----------------

About this guide

Who should use this guide

This *SNA Host Communications User Guide* is intended for Nortel IVR 2.5/S application developers whose voice applications require access to mainframe resources using the SNA communication protocols. In addition, it is assumed that as a Nortel IVR 2.5/S application developer, you are familiar with SNA protocol-based applications, and are experienced in creating voice applications with Nortel IVR 2.5/S. You should also be familiar with the UNIX operating system and the vi text editor or another text editor installed on your application server.

How to use this guide

This guide contains the following chapters:

- Chapter 1: “About host communication links”

This chapter shows the Nortel IVR 2.5/S developer how to build upon the Nortel IVR 2.5/S telephony technology to create communications applications that interchange data with an SNA host.

- Chapter 2: “Installing and configuring the host link software and hardware”

This chapter walks the Nortel IVR 2.5/S developer through the process of setting up the host link.

- Chapter 3: “Creating template files”

This chapter discusses how to use Action and Screen templates.

- Chapter 4: “Using the COM cells to access the host computer”

This chapter shows how to set the parameters for each of the COM cells; it also illustrates a sample application using these cells.

- Chapter 5: “Troubleshooting”

This chapter presents the tools that the Nortel IVR 2.5/S developer can use to gather and debug data. It also contains a list and explanation of error messages.

Additional manuals

The information supplied with your vendor’s host connectivity package provides information necessary for hardware and software installation and configuration.

You may find the following manuals useful while reading this manual:

- *IVR Generator 2.0/S Application Development Guide*
- *IVR Generator 2.0/S System Administration Guide*
- *IVR Generator 2.0/S MRS/RackProduct Manual*
- *IVR Generator 2.0/S MRS/TowerProduct Guide*

Conventions used in this guide

Throughout this guide, several typographic conventions have been used to highlight certain types of information:

- Prompts are shown in a different typeface (for example, `/sci>`).
- Menu names and options are shown in a different typeface (for example, the `sci/install` menu).
- Directories and accounts, are shown in a different typeface (for example, the `nortel` directory).
- Buffer names are shown in all upper-case characters (for example, the `CURRENT MESSAGE` buffer).
- Items that are file names or messages are shown in a different typeface (for example, the `/u/nortel/3270/getbalance.act` file).
- Items you must type are shown in bold in a different typeface (for example, type **te5250** at the prompt).
- Variables shown in command lines appear in italics (for example, the `host_cfgn` file, where *n* is a variable representing a board number).
- Key names you press are shown in bold (for example, the **<F1>** key).

- Keyboards usually have keys named <Return> or <Enter> that perform the same function. For convenience, this guide uses the keyname **<Enter>** to represent both keynames.
- Field names are shown in italics (for example, the *Device* field).

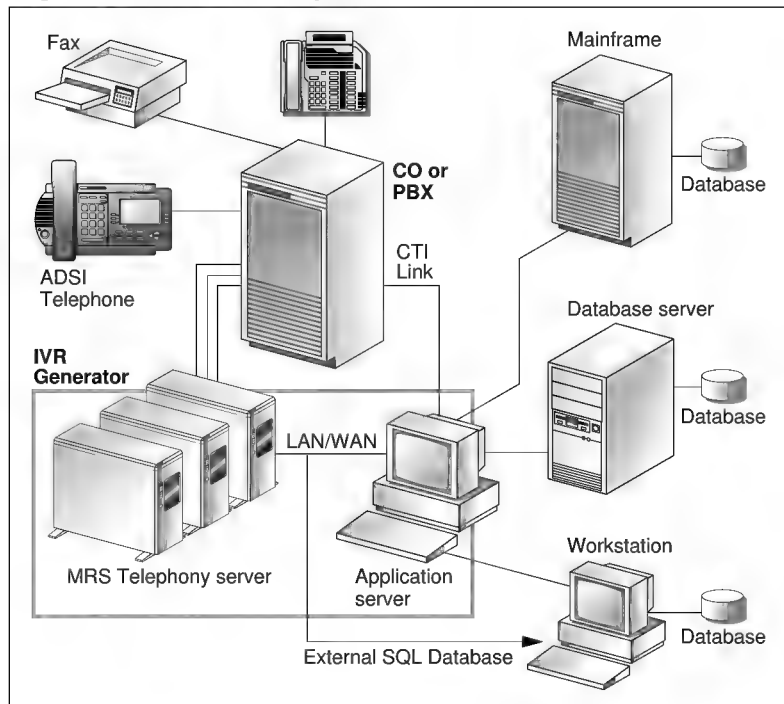
Chapter 1: About host communication links

This chapter shows the process of building upon IVR Generator telephony technology to create communications applications that interact with a System Network Architecture (SNA) host. This chapter contains the following sections:

- SNA gateway hardware and software
- Creating IVR Generator applications that communicate with an IBM host
- IVR Generator COM cells
- TRS
- TRSC
- EXPRESS communications software and hardware

IVR Generator SNA host link communication applications enable users to access information by way of a telephone; this information resides on a host mainframe. You can create IVR Generator applications that automatically and transparently transfer data between a host and a Nortel application server. Figure 1-1 illustrates this flow of data.

Figure 1-1: IVR Generator platform



The SNA Gateway hardware and software

You can install the SNA Gateway on application servers with the SCO 3.0 operating system.

The application server must contain the following:

- synchronous SNA or Token Ring protocol connection to the host computer (see the *Nortel IVR 2.0/S Installation and Maintenance Guide*, Chapter 14, “Host Connectivity Overview,” for more information)
- IVR Generator modules
 - Terminal Resource Server (TRS)
 - Terminal Resource Server Client (TRSC)

Note: For information about hardware specifications, contact your Nortel (Northern Telecom) sales representative.

Figure 1-2: Accessing a host with Operator Assistance on the IVR Generator system

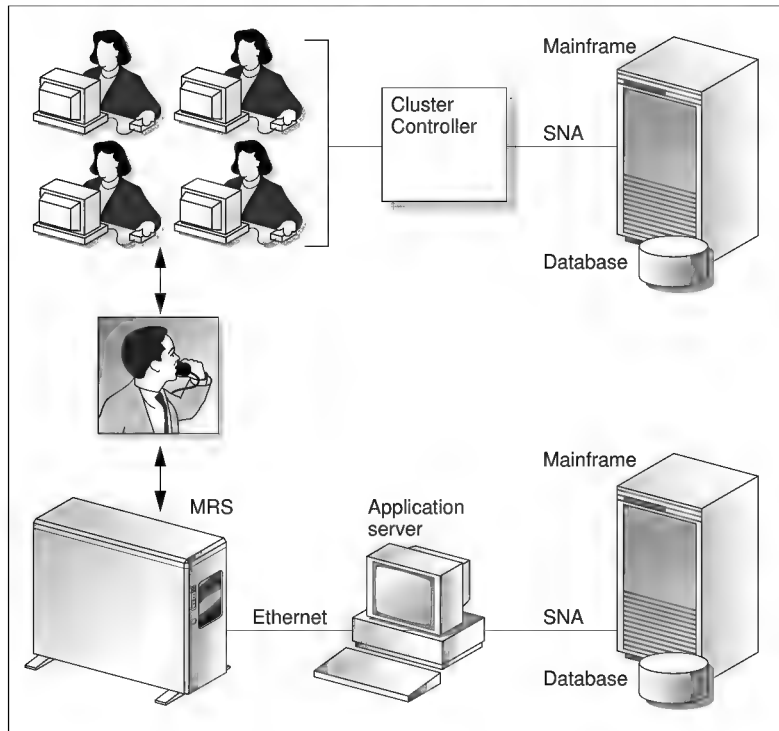


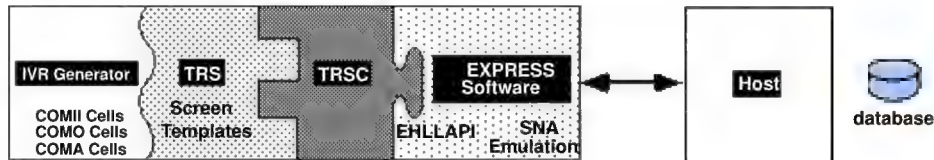
Figure 1-2 shows that a user can call an operator or the IVR Generator system to access information that resides on a mainframe database. The user needs to access the data that is reached through a menu-driven program on a host. An operator must enter and retrieve information stored in the database using an SNA terminal. However, instead of connecting a terminal to the host and retrieving information manually, you can develop an IVR Generator application that logs on to the host, manipulates the host application, and stores the desired information in the buffers of an IVR Generator application. Buffers of information can then be played back to a person using the phone.

Creating IVR Generator applications that communicate with an SNA host

IVR Generator applications communicate with an IBM host by opening connections and sending and receiving data over SNA sessions. This section explains the IVR Generator host link software architecture. Figure 1-3 compares the IVR Generator architectural flow of information to pieces of a puzzle.

This flow begins with an IVR Generator application. The IVR Generator application needs to exchange information with the Apertus EXPRESS software. The exchange occurs through the TRS modules. The application communicates with TRS through the COM cells.

Figure 1-3: IVR Generator SNA gateway software architecture

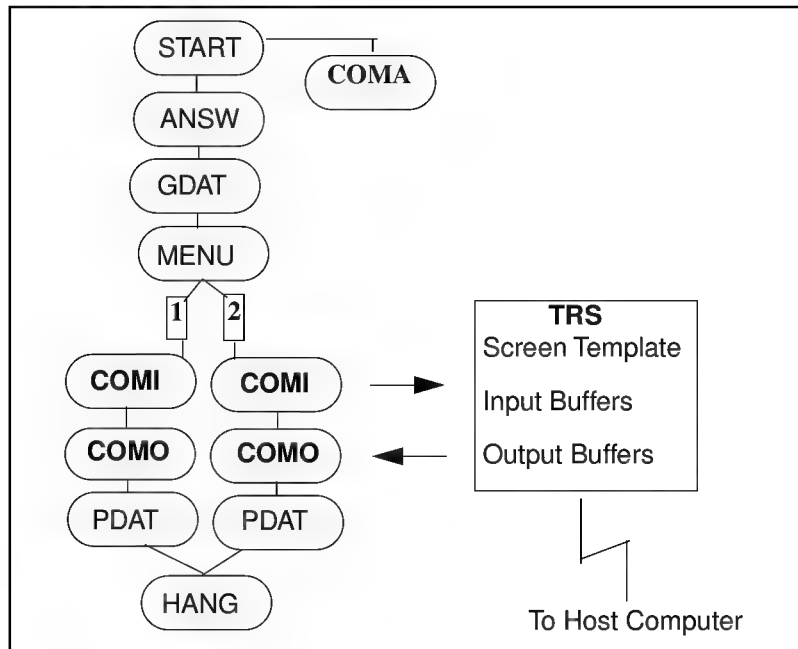


Note: Since the SNA host link process is determined by the IBM environment, the IBM system administrator needs to be aware of the SNA protocol that is being used. This is so that the EXPRESS software can be configured properly.

IVR Generator COM cells

Your IVR Generator applications access the host link through the COM cells. The COM cells provide the call flow interface. You use three types of COM cells to communicate and exchange data with an IBM host: COMI, COMO, and COMA. See Figure 1-4.

Figure 1-4: The COM cells



- The **COMI** cell initiates data exchange by invoking an action template. An action template contains a list of screens that will be traversed to complete the transaction. There are different action templates that handle all contingencies in host transactions. Templates and the procedure for developing templates are described in Chapter 3, “Creating template files.”

Note: A transaction consists of the steps a terminal operator performs in order to complete a task. This includes logging in to the host, navigating through the appropriate screens, and logging out.

- The **COMO** cell collects host data and places this retrieved information into buffers. Any IVR Generator cell can then use the information.
- The **COMA** cell cancels a host transaction. Typically, the **COMA** cell is executed if there is a problem with the host transaction.

Chapter 4, “Using the COM cells to access the host computer,” shows how to set the **COMI**, **COMO**, and **COMA** cell parameters.

To use TRS, place the COM cells in the IVR Generator application call flow at the point where you need to set up a host transaction or data exchange.

TRS

An IVR Generator process called the Terminal Resource Server (TRS) controls all SNA host link sessions and interfaces with the installed terminal emulation package. In addition, TRS manages all host connections. The TRS runs as a stand-alone process within the IVR Generator architecture and starts automatically at IVR Generator startup. The TRS and TRSC files are located in `/u/nortel/exe`.

Note: During the IVR Generator 2.00.12 upgrade procedures, if you select “3” (none) at the prompt “Please select host communication vendor []:”, then you must do the following steps:

- 1 Copy the original TRS and TRSC files in order to preserve these files.
- 2 Rename TRS_SSI to TRS in the `/u/nortel/exe` directory.
- 3 Rename TRSC_SSI_4153 to TRSC in the `/u/nortel/exe` directory.

Action and screen templates

The TRS process uses template files to control the communication process between the application and the host computer. The template files provide a way of understanding the screen images in a host application. These files reflect the location of the input and output fields on the host screens. For details on creating screen templates, refer to Chapter 3, “Creating template files.”

TRSC

The Terminal Resource Server Client (TRSC) communicates with the specific vendor communication package. TRSC takes data from the TRS and formats it for use with the specific vendor hardware and software.

EXPRESS communications software and hardware

The EXPRESS communication multiprotocol adapter board and EXPRESS communications software manage the actual host connection which transmits and receives data that complies with the SNA network protocols.

EHLLAPI

Using the IBM-defined Extended High Level Language Applications Programming Interface (EHLLAPI), you can search through the host screen data for specific strings of text, wait for specified actions, and perform other functions. You set up these transparent functions in the screen templates. This process is also referred to as screen scraping.

Terminal emulation

Terminal emulation occurs at the remote sites using standard SNA protocol terminal emulators. These terminals give multiple users access to resources residing on a host computer. The terminals use the specific protocol (3270 or 5250) for displaying information on their screens and returning screen contents to the IBM host. Each field on an input screen contains unique attribute information. Attributes identify the characteristics of the information written to the screen, such as highlighted text and reverse video.

The SNA communication adapter uses the same technology. An adapter installed in an application processor can emulate many concurrent SNA protocol terminal sessions over a single synchronous connection. By complying with the SNA communication protocols, IVR Generator applications can support multiple logical SNA terminal sessions. Physical terminals are not required. Your IVR Generator applications can access the same host resources as operators using SNA protocol terminals.

For example, consider a menu-driven program on a host. This program allows an operator to enter and retrieve information stored in a database. Instead of connecting a terminal to the host and retrieving information manually, you can develop an IVR Generator application that logs on to the host, manipulates the host application, then stores the desired information in the buffers of an IVR Generator application. Buffers can then be played to a person using the phone.

Optimally, your application can simultaneously run multiple terminal sessions and process different transactions for different users.

Chapter 2: Installing and configuring the host link software and hardware

This chapter discusses the process of setting up host link communications. The sections of this chapter are presented in the order you follow to successfully establish the communications:

- Site planning
- Installing the communications hardware
- Installing the EXPRESS Software
- Starting EXPRESS
- Configuring EXPRESS
- Verifying connections to the host computer
- Configuring IVR Generator host communications
- Using the start-up script
- Bringing up IVR Generator including TRS

Site planning

Review the following sections, “Pre-site planning” and “Package contents,” before you conduct installation and configuration procedures outlined in this chapter.

Pre-site planning

Before starting the installation and configuration procedures in this chapter, determine the required physical configuration of the installation site:

- communications protocols
- number of logical units (LUs)
- communication media (SDLC or token ring)
- host information
 - LUs created/working
 - Ports created/working
 - Host MAC address (Token Ring)
 - Secondary station address (SDLC)
 - Network ID
 - XID block
 - Control point
 - Remote MAC address (Token Ring)

Package contents

Verify that you received the package you ordered:

- correct number of logical units
- correct version
- documentation

Installing the communications hardware

The communication hardware installation procedure consists of

- installing the communications adapter in your application server
- connecting the communications adapter to your physical SDLC or Token Ring communications media
- verifying that the hardware is correctly integrated with your application server

For this procedure, refer to the documentation that comes with the specific communication adapter you are installing.

The SCO platform requires the following information:

- Interrupt Level (IRQ)
- I/O Address
- Memory Address
- DMA Channel (token ring)

Installing the EXPRESS software

EXPRESS is an integral part of the host link. The EXPRESS software package, coupled with a communications adapter, handles all of the lower-level SNA communication. Once correctly installed and configured, this package transparently maintains an SDLC or Token Ring connection with the host.

The communications adapter provides the physical connection between your application server and the SDLC or Token Ring host connection. You install the communication adapter in an expansion slot in your application server.

EXPRESS is a product of Apertus. This company provides a complete documentation set that covers the installation and configuration of EXPRESS. You need the following documents in order to install the EXPRESS software:

- *EXPRESS Installation Guide* (for your specific platform)
- *EXPRESS Release Notes*

You need the following EXPRESS guides for referencing details of the SNA interface configuration and operation:

- *EXPRESS New Features Guide*
- *EXPRESS SNA Server Configuration Guide*
- *EXPRESS Desktop Configuration Guide*
- *EXPRESS Administrator's Guide*
- *EXPRESS User Interface Guide*

The EXPRESS Server and Desktop

EXPRESS consists of the following components:

- EXPRESS SNA Server
- EXPRESS Integrated Desktop
- EXPRESS API Software Development Kit
- ESS API Runtime

Of these, you install only the EXPRESS SNA Server and the EXPRESS Integrated Desktop. These are the components essential for SNA communication. The Server performs the actual terminal emulation. The Desktop is the user interface to the Server. You use the Desktop to configure the Server.

EXPRESS installs the Desktop automatically when you install the SNA Server.

During installation, the EXPRESS installation script refers to the Server and Desktop as one package called exsrv+dsk.

Installing the EXPRESS SNA Server and Desktop

Note: Log in to your application server as **root** before you begin any installation procedures.

To install the EXPRESS software on your system, refer to the *EXPRESS Installation Guide* and the *EXPRESS Release Notes*. During the installation procedure, you are asked to respond to the questions in Table 2-1.

Table 2-1: Installation prompts

Installation prompt	Attribute	Default	Guidelines
Is this a stand-alone or networked environment?	Required	stand-alone	Choose <code>stand-alone</code> if all EXPRESS software will be installed on only one system which contains the communications adapters.
Enter the name of the EXPRESS runtime directory for this system.	Required	System Dependent	Enter the path name of the directory where you want to install the runtime modules. The EXPRESS runtime directory is used as part of the path name for the default standard backup directory.
Do you want EXPRESS to start at system boot time?	Required	Y	Select <code>Y</code> to start EXPRESS automatically at boot time. If you select <code>N</code> , you must start EXPRESS by entering <code>express_adm start</code> .
Do you wish to add the EXPRESS environment variables to the <code>/etc/profile</code> ?	Required	Y	Add the environment variables to <code>/etc/profile</code> to ensure all EXPRESS users have the proper environment to run EXPRESS.

Indicate the protocol drivers you need to interact with IVR Generator:

- SNA over Token Ring
- SNA over SDLC
- LU6.2/APPN (for 5250 communications)

EXPRESS Server installation procedure

This section lists the steps for installing the EXPRESS SNA server. For more detailed explanations of this procedure, refer to the *EXPRESS Installation Guide* and the *EXPRESS Release Notes* for your platform.

Procedure 2-1: EXPRESS server installation

- 1 Log in as **superuser** at the computer system where you are installing the EXPRESS software.

- 2 Ensure that you have a bootable backup of the UNIX kernel.

- 3 Start the installation script by entering the following command:

```
pkgadd -d device exsrv+dsk 2>&1 |tee pkgadd.out
```

- 4 Answer the prompts for the EXPRESS environment questions.

- 5 Select all the EXPRESS protocol drivers you need.

- 6 Select the communication adapter(s) to install and select the configuration parameters for these adapters.

When you are finished selecting the communication adapters, the installation script informs you of the choices made during the installation process and the path to the `sysconfig` file that contains the list of your answers.

- 7 Review your choices:

- If you have made the correct choices, continue the installation procedure.
- If you made an error, rerun the installation.

- 8 If necessary, tune the UNIX kernel parameters.

The installation script sets values for kernel parameters based on an assumed maximum number of users and sessions in EXPRESS. To ensure appropriate use of your system resources, you may need to adjust the kernel parameter values. Refer to the *EXPRESS Release Notes* for the values that the installation script sets for kernel parameters related to EXPRESS.

- 9 If necessary, reboot the system following the procedure defined for your UNIX system.

Files affected by EXPRESS installation

Table 2-2 lists files that EXPRESS “touches” during installation.

Table 2-2: Files affected by EXPRESS

File	What is touched
/etc/profile	Environmental Variables
/etc/services	TCP port used by EXPRESS
/etc/passwd	Sets up the user “express”
/var/EXPRESS/envfile	Kernel-related environmental variables

Starting EXPRESS

Now that you have installed EXPRESS, launch it with the following commands.

Procedure 2-2: Starting EXPRESS

Note: Your terminal must be configured before you run EXPRESS software. For information on configuring your terminal, see the *Nortel IVR 2.0/S Installation and Maintenance Guide*, Chapter 15, “3270 over SDLC software installation and configuration”, page 15-5.

When your terminal has been configured, follow these steps:

- 1 As `superuser`, enter

```
. /etc/profile
```

```
express_adm start
```
- 2 Start up EXPRESS as a user. You do not need to be a superuser to start the EXPRESS user interface. Enter

```
express -u express
```

The EXPRESS Manager appears. This tool handles configuration functions for the administrator or designated user.

See the *EXPRESS SNA Server Configuration Guide* for information on navigating through the screens.

Configuring EXPRESS

EXPRESS configuration includes the hardware, communications, and user configuration. It involves creating a configuration profile that is stored in the EXPRESS database. The complete set of hardware, communication, and user profiles is called a configuration. EXPRESS allows the definition and maintenance of multiple configurations. However, only one configuration can be active at a time.

This section provides an overview of the EXPRESS configuration process. For a detailed discussion, refer to the *EXPRESS SNA Server Configuration Guide*.

Defining hardware

EXPRESS supports communication adapters, computer systems, and ports. An EXPRESS configuration must contain information about all computer systems and communications equipment in the domain. Most of this information is defined during the EXPRESS installation procedure.

Defining communications

The configuration process must define the various objects that EXPRESS uses to make a connection with remote links such as data links, stations, and logical units. The EXPRESS SNA tree is the main tree of the Define Communications task. It displays a hierarchy of communication objects. The first step requires you to define a node. Refer to the procedure on defining nodes in the *EXPRESS SNA Server Configuration Guide*.

Note: For further information on defining nodes, see the procedure called “Defining the node” in your respective host connectivity software and installation chapter in the *Nortel IVR 2.0/S Installation and Maintenance Guide*.

Establishing a naming convention

When creating nodes, links, stations, and LU pools during the configuration portion of this chapter, you should use a naming convention that accurately describes the nodes, links, stations, and LU pools. The naming scheme requires defining a prefix that indicates the host port, and then adding the suffix “link” or “station” to the prefix. The hierarchy of nodes, links, stations, and LU pools follows:

Node
 AceLink
 Acestation
 AcePool
 BTWLink
 BTWStation
 BTWPool
 CBELink
 CBE1Station
 CBE1Pool
 CBE2Station
 CBE2Pool

The EXPRESS configuration procedure echoes this structure. You configure EXPRESS in this order:

- Define a Node.
- Define Links.
- Define LUs.
- Define Stations.

Refer to the EXPRESS documentation for detailed configuration procedures for nodes, links, LUs, and stations.

Defining SDLC links and stations

After you define an SNA node, you can insert an SDLC data link under any SNA node in the EXPRESS SNA tree. Refer to the “Defining SDLC” section in the *EXPRESS SNA Server Configuration Guide*. The default configuration parameters can be used for the link, except for the DSR Wait Timeout which you should decrease from 600.0 (10 minutes) to 60.0 (1 minute).

After defining the link, add a station under the link. Most of the station parameters depend on values required by the remote host. The parameter “Restart connection if it fails” should be set so that the station keeps trying to connect if the remote systems goes down.

Note: For more information on defining SDLC links and stations, see the procedures called “Defining the Link” and “Defining the station” in the chapter for your respective host connectivity over SDLC software installation and configuration, in the *Nortel IVR 2.0/S Installation and Maintenance Guide*.

Defining Token Ring links and stations

After you define an SNA node, you can insert a Token Ring link under any SNA node in the EXPRESS SNA tree. Refer to the “Defining Token Ring Links” section in the *EXPRESS SNA Server Configuration Guide*.

After defining the link, add a station under the link. Most of the station parameters depend on values required by the remote host. The parameter “Restart connection if it fails” should be set so that the station keeps trying to connect if the remote systems goes down.

Note: For more information on defining Token Ring links and stations, see the procedures called “Defining the Link” and “Defining the station” in the chapter for your respective host connectivity over Token Ring software installation and configuration, in the *Nortel IVR 2.0/S Installation and Maintenance Guide*.

Defining LU pools (3270 only)

Each station requires one or more pools of LUs that can be assigned to individual sessions. An LU pool consists of a group of LUs that all have the same parameters. The addresses for the pool can be any addresses between 2 and 255. Use “-” to define a range of addresses (for example, 2-10). Use “,” to list all sets (for example, 2-10, 12, 15-255).

Defining LU 6.2 (5250 only)

When you define an SNA node, an independent LU Type 6.2, the Control Point LU, is automatically created. For detailed procedural information, refer to the “Defining LU 6.2s” section in the *EXPRESS SNA Server Configuration Guide*.

Defining LU6.2 consists of three distinct tasks:

- defining remote LUs
- defining modes
- defining the 5250 symbolic destination

Defining remote LUs

The process of defining a remote LU provides EXPRESS with the specific address and characteristics of one host LU. Define each LU that you want the TRS to use. EXPRESS associates three attributes to each LU which you define. These attributes are Remote LU, LU Pair, and Associating Mode. You define these attributes in the EXPRESS hierarchical structure.

Remote LU

To define a Remote LU, provide EXPRESS with your SNA network ID, the host's control point, and the address of the LU. Obtain this information from your source of the host information. You also need to name the Remote LU. This name allows you to refer to the LU by name instead of address. Choose a meaningful name, one that allows you to make an association with the Remote LU. You can number the names. For example, the LU name "Banking LU1" would be appropriate if you are connecting to a banking host application.

LU pair

An LU Pair maps the LU you defined in the previous section, "Remote LU," to your Control Point LU. You can specify remote LUs, local control points, and security characteristics. When naming the LU Pair, mention the control point LU. For example, if the Control Point LU is named IVRComPt, use this name for the LU Pair.

Associating mode

Associating a mode provides specific information about your host connection at the LU level. The most important characteristic of the mode is that it matches the mode that the host expects you to use. EXPRESS offers several modes from which you can choose. You can also define a mode. See the "Defining a mode" section that follows.

Defining a mode

Defining Mode allows you to create an appropriate mode for your host connection if the default modes provided with EXPRESS are inadequate. You obtain this mode name from your source of the host information.

Defining a 5250 symbolic destination

The 5250 Symbolic Destination ties together your Remote LU, your Mode, and the Control Point LU. Use this when you define a session. Refer to the "Defining Users and Sessions" section in this chapter.

Defining users and sessions

The EXPRESS user/session configuration defines the EXPRESS users, and the sessions and supervisory tasks available to these users. Configuration of users and sessions varies with each user population. Refer to the “User and Session Planning” and the “Defining User and Session Profiles” sections in the *EXPRESS SNA Server Configuration Guide*. Also, the *EXPRESS Integrated Desktop Configuration Guide* provides comprehensive material on defining users for the 3270 and 5250 communications environments.

The IVR Generator TRS process handles eight sessions for each EXPRESS user. The first user needs to be named “express” to map to the user created during EXPRESS installations. To determine the number of users to define, divide the total sessions required by eight. All sessions are named “a” through “h.” For the LU address, enter the LU pool name.

Setting limits for terminal emulation

The final step in the EXPRESS configuration process requires setting limits for the terminal emulation service. Refer to the *EXPRESS Desktop Configuration Guide* for details about setting limits for this service. You need to set limits for two of the three services listed: the standard service and the service your system names. For both of these services, set “maximum users” and “EHLAPI user programs” to the number of users you defined.

Verifying connections to the host computer

Before attempting to bring up the entire host link and run an IVR Generator application, you should test each part of the host link separately. This makes it easier to locate problems since you are dealing with a small part of the host link instead of the entire configuration.

This section covers testing to make sure EXPRESS can connect to and interact with the host.

Checking the status of your links and stations

The first step requires bringing up the EXPRESS Manager and checking the status of your Stations and LUs.

Procedure 2-3: Checking the status of your stations and LUs

Note: Your terminal must be configured before you run EXPRESS software. For information on configuring your terminal, see the *Nortel IVR 2.0/S Installation and Maintenance Guide*, Chapter 15, “3270 over SDLC software installation and configuration,” page 15-5.

When your terminal has been configured, follow these steps:

- 1 Display the EXPRESS Manager by entering the following:

```
express -u express
```

This displays the EXPRESS Manager Main Window.

- 2 Click **Control** and **Administration**.

This displays the Control and Administration window.

- 3 Click **Control Communications** and **Diagnostics**.

This displays the Links, Stations window which shows link and station status. In this window, make sure all of your links and stations have status enabled.

If a link or station is not enabled, do the following:

- a Single click on the link or station.
- b Under the **Control** pulldown menu, select **Activate**.
- c Make sure the status changes from “Inactive” to “Enabled.” If the status is “Pending,” the system is waiting to contact the host.

Bringing up an interactive session with the host

Bringing up an interactive session with the host is probably the best way to check your host connection because you see output from the host and you can actually interact with the host. If IVR Generator is running on your system, you should bring it down to ensure that all sessions can interact with EXPRESS.

Then, from the EXPRESS Manager, click *Interactive Sessions*. This brings up the *Interactive Session* window. This window displays all sessions. You may only select a session that is *not* in use by IVR Generator. Click the session you want to use.

This brings up the sample SNA session window.

If you are connected to the host, this window displays the host's main or introductory screen.

If you are not connected to the host, the window remains empty.

Configuring IVR Generator host communications

In order to integrate the EXPRESS software with the IVR Generator TRS process, two configuration files define the EXPRESS configuration to TRS and map EXPRESS LUs to application programs and templates.

Setting up the `trs.conf` file

The `trs.conf` file specifies the Initial-Action template for each session you define on the IVR Generator application processor. The `trs.conf` file must reside in the `/u/nortel/3270` directory. The following example shows the syntax for the `trs.conf` file:

```
app-name:board-number>session-number:initial-template:hear  
tbeat:protocol
```

The colons (:) and the greater than (>) symbols signify field delimiters and must be placed in the specified positions without any additional white space.

app-name

The `app-name` entry identifies the mainframe application to which you are assigning sessions. The name entered here is the same name you enter in the action templates that are executed by the IVR Generator application.

board-number

The board-number entry should be 0 for the first (or only) board, and 1 for the second.

session-number

The session-number entry determines the addresses of LUs to activate at startup. Enter it as a range of addresses. The addresses must range from 2 to 255 (for example, 2–33).

initial-template

The initial-template entry identifies an Initial-Action template (without the .act file extension) for setting the start-up action for the specified sessions when connecting to the host computer. The start-up action brings the specified sessions to the screen on the host computer where the action templates start. Action templates are described in detail in Chapter 3, “Creating template files.”

heartbeat

You can specify an optional heartbeat action for the application specified by app-name. You can use this feature to send an indication to the host that a session is still active. Some hosts log out sessions that remain idle for a period of time. You can also use the heartbeat action to check connectivity, and to verify that the session remains on the appropriate screen. Typically, the heartbeat action contains a Logout-Action template that brings the session back to the login screen if the connectivity with the mainframe fails. If you do not need a heartbeat action, enter a hyphen (-) for this action. For additional information, refer to Chapter 3, “Creating template files.”

protocol

The protocol entry indicates the communications protocol being used by app-name. This field is required and should be set to 3270 or 5250.

Example of a trs.conf file

Consider the following example. Initial-Action template files login.act and signin.act have been defined and reside in the /u/nortel/3270 directory. There are four applications that you want to access on the host computer:

- accounting, accesses the accounting software
- market, tracks stock market activity

- banking, retrieves credit balances
- airline, for purchasing tickets on a local commuter carrier

The application names shown here are not necessarily the actual names assigned on the host computer, but they are the names assigned on the application server for the `trs.conf` file and all action templates that use these applications.

You want to set up the `trs.conf` to initialize the sessions as follows:

- Initialize sessions 2–3 for accounting with `signin.act`.
- Initialize sessions 4–8 for market with `login.act`.
- Initialize sessions 9–10 for banking with `login.act`.
- Initialize sessions 15–17 for airline with `signin.act`.

The following example illustrates a specific `trs.conf` file:

```
#trs.conf file to set up sessions for the application
server
accounting:0>2-3:signin:ping@60:5250
market:0>4-8:login:-:5250
banking:0>9-10:login:-:5250
airline:0>15-17:signin:ping@60:5250
```

Setting up the `trs.sdf` file

The `trs.sdf` file maps the short name of each session to the LU number. TRS requires the `trs.sdf` file when using the Apertus EXPRESS host connectivity package. See the documentation accompanying the EXPRESS package for information on configuring EXPRESS for your environment. Configuring EXPRESS provides you with the values for the `trs.sdf` file. The `trs.sdf` file follows this format:

U:Username	
Session Short Name	LU Number
Session Short Name	LU Number
Session Short Name	LU Number
•	•
•	•
•	•

You can list up to eight sessions and LUs under each EXPRESS Username. The following is an example of a `trs.sdf` file:

```
U: express
a 2
b 3
c 4
d 5
e 6
f 7
g 8
h 9
U:expressa
a 10
b 11
c 12
d 13
e 14
f 15
g 16
h 17
U:expressb
a 18
b:19
"trs.sdf" 23 lines, 140 characters
```

Specifying separate IDs/passwords for each LU

If your host application requires that each session use a separate login and password, list as many different logins and passwords as you plan to use in the `lubuf.dat` file. If your host application only needs a single login ID and password for each session, place the login ID and password in a screen template. In the single ID case, you do not need to create the `lubuf.dat` file; just use the screen template directly. The `lubuf.dat` file must reside in the `/u/nortel/3270` directory. It may contain up to four variables for each individual session, two of which are the `LOGIN_ID` and the `PASSWORD`. If you require fewer than four variables, leave the unused variables blank.

The field definition for `lubuf.dat` follows:

```
board# session# &LOGIN_ID &PASSWORD &LU_BUF1
&LU_BUF2
```

Each field should be separated by space(s) or the TAB key. Each line of the file specifies the contents of defined variables for each particular session number. In most cases, hosts only require one login; therefore, you typically do not specify the `&LU_BUF1` and `&LU_BUF2` variables. For example, the contents of the `lubuf.dat` file could look like the following:

```
0 2 vtek001 voice1
0 3 vtek002 voice2
```

The first line of the file indicates that the contents of the `&LOGIN_ID` variable and `&PASSWORD` for board0 session 2 are “vtek001” and “voice1”. In the previous example, only two variables are needed; therefore, you do not need to enter anything for variable `&LU_BUF1` and `&LU_BUF2`.

To make TRS use the `lubuf.dat` file, the screen template must indicate that multiple `login_id` and `password` entries exist by placing the ampersand before the `LOGIN_ID` entry and the `PASSWORD` entry as indicated below in the sample screen template:

```
screen_name 1,2 validation_tag
2,20 - &LOGIN_ID
3,20 - &PASSWORD
```

Using the start-up script

Nortel provides a template for a script that automates the EXPRESS start-up background processes and enables your configuration's links and stations. Name this script `express.start` and place it in the root `(/)` directory. This template must be modified to match your environment and configuration.

```
#!/user/bin/sh
#####
#
# Customizable option: shell
#
./user/lpp/express/.profile

#
# Customizable option: profile
#

#
MESSAGE1="\nInvalid request usage:manager (start |
stop)"
MESSAGE2="\nTimeout occurred. Necessary daemons
not active after 5 minutes"
MESSAGE3="\nAll daemons now running, executing
```

```
express start"
#
#check to see an action parameter (start or stop)
#was passed in
#

if["$#" -lt 1]
then
echo$MESSAGE1
exit 1
fi
#
#start the express background processes
#

express_adm start

#
#loop to verify all necessary daemons are running
#then start express in hidden mode
#
loopCount=0
retryCount=0
#
#hang out in a loop until all express daemons are
#running. This process will time out after 5
```

```
minutes
#

while [ $loopCount -eq 0];do
  (ps -ef | grep cdm | grep)
  && (ps -ef | grep agent | grep -v grep)
  && (ps -ef | grep jobex | grep -v grep)
  && (ps -ef | grep executive | grep -v grep)
  && (ps -ef | grep watchdog | grep -v grep)

  #
  # Not applicable for SCO
  #
  if [ $? -ne 1 ]
  then

    echo $MESSAGE3
    break
  else
    retry Count=`expr$retryCount + 1`
    if [ $retryCount -eq 12]
    then
      echo $MESSAGE2
      exit 2
    else
      sleep 5
    fi
  fi
done
case "$1" in
  'start')
    #
    # Customizable option: Name of token ring.
    # Comment out if this is a serial connection.
    #
    stst<<!
      start datalink trlink
    !
    #
    # Customizable option: Name of token ring station.
    #
    stst<<!
      start connection trstation
    !
  
```

2-22 Installing and configuring the host link software and hardware

```
#
# Customizable option: Name of express users.
#
    express -u express -h
    express -u express0 -h
    express -u express1 -h
    express -u express2 -h

                                ;;
'stop')
    echo "STOP Session Managers"
    express -x -u express
                                ;;
esac
```

Editing system files (for SCO only)

EXPRESS requires you to edit the `/etc/rc2.d/S99 express` file. Add the following lines:

```
if [ $1 = start]
then
rm -f/u/express/db/cdmst
fi
su root -c "/u/express/bin/express_adm $1"
su root -c "/express.start $1"
```

Note: The paths specified in these lines may differ at your site.

Bringing up IVR Generator including TRS

Start-up switches

TRS uses the switches listed in Table 2-3 to modify specific operations.

Table 2-3: TRS start-up switches

Switch	Purpose
q	Message ID used by process. For IVR Generator Version 2.00 or greater, this value is 98; otherwise, it is 97.
v	Verbose flag. Causes output to be generated to <code>trs.log</code> and <code>trs.tmp</code> .
r	Revision flag. Displays the current trs version.
c	Check flag. Displays information on the trs version and verifies the correctness of the application templates and action files.
b	Background mode. Application default.
e	Envoy mode.
t	Transparent mode. Ignores control characters.

Testing communication with IVR Generator

Use the procedure in this section to ensure that IVR Generator can interact with EXPRESS.

Prerequisites

This procedure requires instructing the host link to run in verbose mode. In addition to confirming that the host link and IVR Generator can communicate, it also provides significant data about how the host link is performing. This information assists in isolating host link problems.

Procedure 2-4: Testing interaction between IVR Generator and EXPRESS

Note: This procedure requires an experienced UNIX user. See your SCO UNIX documentation for information on using the vi editor.

In the IVR Generator home directory, edit the start-up file as follows:

- 1 Change the line from `./trs -b`
to `./trs -b -v`

You can “tail” the file `trs.log` by issuing the command **`tail -f trs.log`**

- 2 If it is not already started, start the host link .
- 3 Reset IVR Generator.
- 4 Load and run an application that utilizes host connectivity.

The file `trs.log` begins to fill with information. This information indicates communication between IVR Generator and the host link. The information in this `trs.log` provides feedback for debugging and troubleshooting. See Chapter 5, “Troubleshooting.”

Chapter 3: Creating template files

The TRS uses action and screen templates to maneuver through the screens of a host application. These templates provide the host with the same input as a terminal operator. The templates also receive output from the host.

Note: Unprintable characters received from the host are converted into spaces using the `-t` parameter.

The application development process requires creating these templates. Before creating the templates, you need to understand the transactions that occur in the host application.

This chapter explains how to do the following:

- Determine the actions a terminal operator performs to enter and retrieve information.
- Create the different action templates that define the sequences of host application screens for each transaction.
- Create the screen template files that define the sequence of fields encountered on each screen.

Determining the required transactions

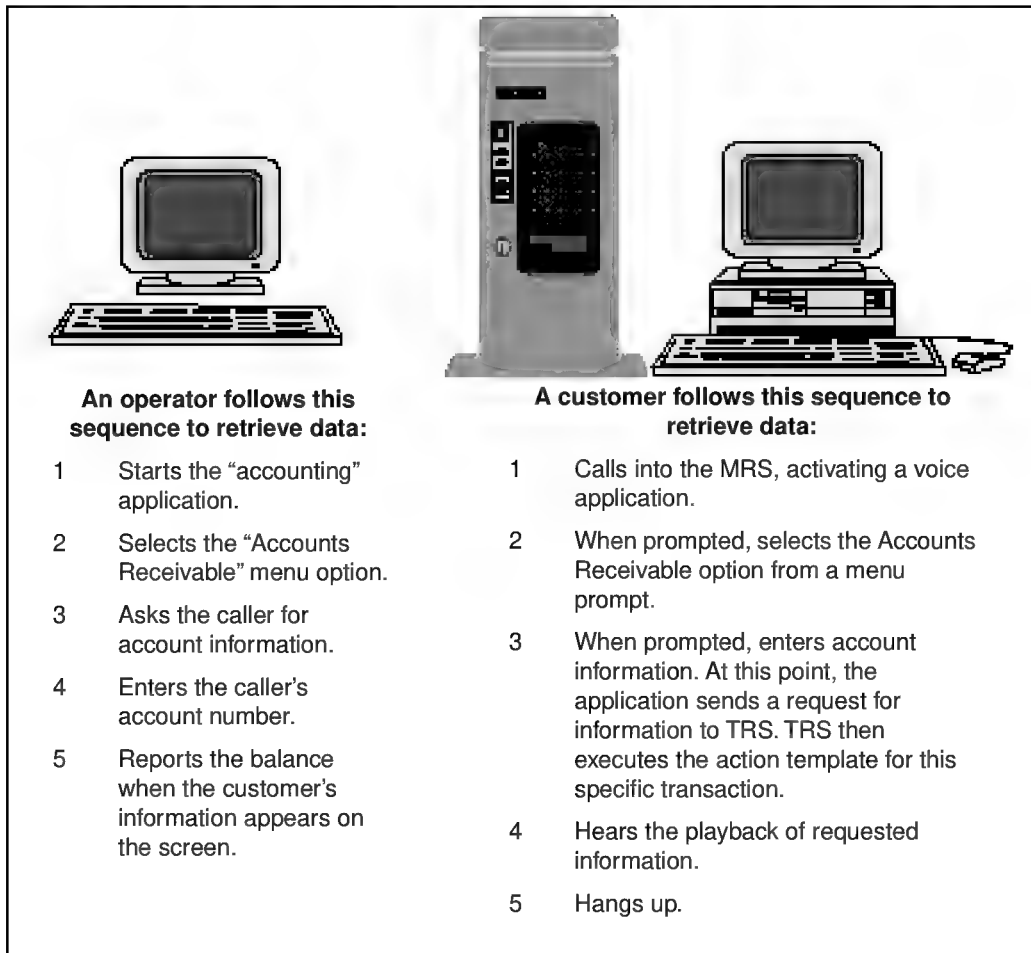
Imagine that you are an operator sitting at a terminal. In order to perform a specific task, you type information and press function keys until you accomplish the desired task. Perhaps a caller asks you to look up a customer's account balance, or enter an order. The series of steps that you perform at the terminal enables the application on the host computer to complete the transaction.

3-2 Creating template files

When an operator at a terminal performs a transaction, the person types information into the specific fields at specific locations on a screen. The operator first must log in to the system. Then, the operator can perform many transactions before logging out.

To develop a voice application that accesses the same information as a terminal operator, you need to tell IVR Generator how to execute the same series of actions that the terminal operator executes. You provide this information in ASCII files called screen and action templates.

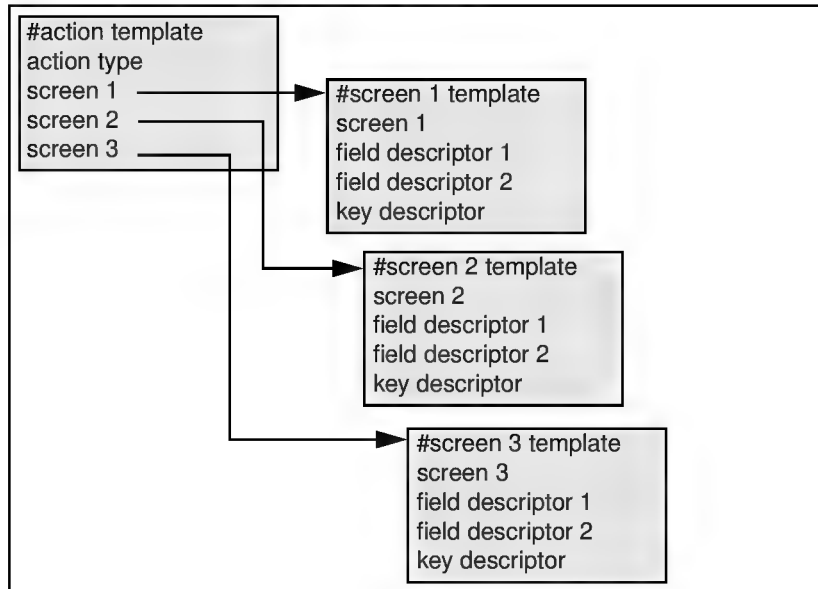
The screen and action template files provide the layout and content of each screen in the host application as the terminal operator sees it. Figure 3-1 compares a transaction done by a terminal operator to one done by a customer calling in to a voice response system.

Figure 3-1: Sample transaction—Terminal Operator versus a Voice Response System

Action templates describe the sequence of screens traversed in order to perform a specific transaction. Screen templates validate each screen, define the fields on the screen that require data, and define all keystrokes required for the screen.

Figure 3-2 shows how screen templates relate to action templates.

Figure 3-2: Action and screen templates



SNA-based applications often contain format inconsistencies that make it difficult for TRS to efficiently determine when the host application is ready for input and what region of the host screen contains vital information. These inconsistencies are due to the character-based format of the SNA protocols.

In addition, the SNA communication protocols cannot notify TRS that host data transmission ended. However, the terminal operator knows when the host transmission ends because the requested information appears on the screen. From the TRS perspective, nothing in the SNA protocols provides notification that host output is completed. The TRS encounters a stream of data from the host and, using this stream, it determines the identity of each screen in addition to locating the end of each screen. To enable TRS to do this, in addition to handling screen inconsistencies, you create a file called the `trs.conf`.

The action templates, screen templates, and `trs.conf` file are ASCII text files that use a simple syntax to define the screen flow and input/output fields. The sections that follow provide a detailed explanation of the templates, as well as the information necessary to create the `trs.conf` file.

Overview of action templates

The host link uses action templates to navigate through the different screens of a host application. The action templates describe the order in which the host link encounters host screens. There are several types of special action templates, as described below.

Each type of action template is suited to a specific portion of a transaction. Each of these action templates can handle several host screens.

Initial-Action templates

The host link executes the Initial-Action template automatically at IVR Generator startup. This template handles login and navigates to the screen from which all regular action templates start. See the “Initial-Action Templates” section in this chapter for the procedure to create Initial-Action templates.

Action templates

The regular action templates start where the Initial-Action Template leaves off. Create one action template for each transaction. See the “Action Templates” section in this chapter for the procedure to create regular action templates.

Reset-Action templates

The Reset-Action template moves a completed transaction back to the screen from which an action template can perform another transaction. See the “Reset-Action” section in this chapter for the procedure to create Reset-Action templates.

Logout-Action templates

In the example in Figure 3-2, the Logout-Action template can recognize any screen in the host application and, from that screen, can move the transaction back to the first login screen. Then, the Initial-Action template can take control.

All action templates operate on the principle that the host screens appear in a consistent sequence, and that each screen is formatted consistently. If the host application presents screens in an erratic manner, or if screens are not formatted consistently, you may have difficulties executing your templates. If this occurs, your Action or Reset-Action templates may stop on an unexpected screen, or the LU may enter a locked state. For these reasons, you should create a Logout-Action template. Refer to the “Reset-Action Templates” section in this chapter for this procedure.

Keep-Alive/Heartbeat template

You can specify an optional Keep-Alive or Heartbeat action for a transaction. You can use this feature to send an indication to the host that a session is still active, since some hosts log out sessions that remain idle for a period of time.

Creating action templates

The first step in creating action templates is to gain an understanding of the host application. To do this, you need to access the host application through a terminal or an EXPRESS interactive session (see Chapter 2, “Installing and configuring the host link software and hardware”). Navigate through the host application the same way you want your IVR Generator application to navigate. You may want to consider taking notes or creating a flowchart of the screens you encounter.

When examining the host application, analyze it in terms of transactions. In other words, note the screens you encounter when performing a specific transaction from start to finish.

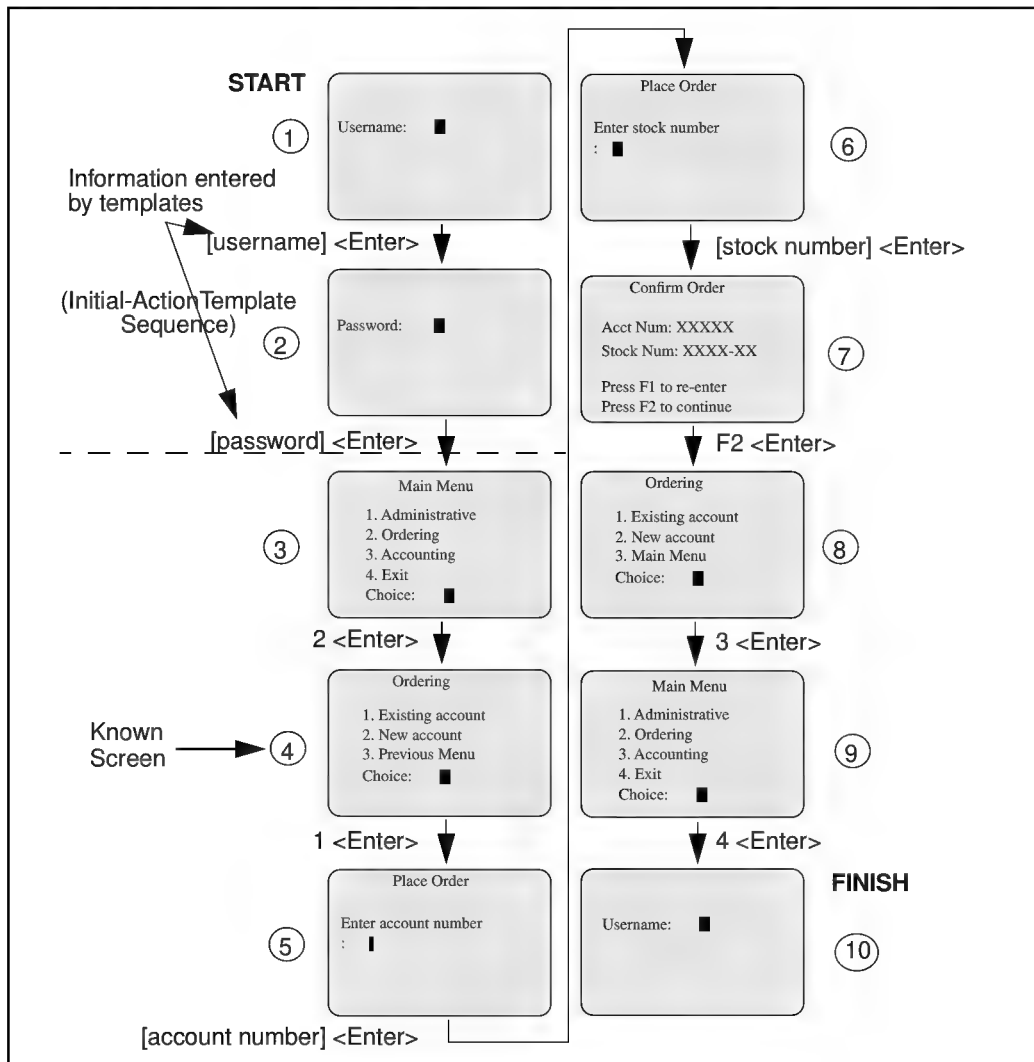
You may want to designate one of the screens of the host application a “Home” or “Known” screen. This should be a screen encountered early in the host application, and on one that the application always ends. The known screen occupies an essential role during the creation of some of the specialized action templates.

Note: The Initial-Action template typically logs you in and displays at the known screen.

An example of a host application follows. Although it is based on a simple host application, this example demonstrates the principles you must understand to effectively create action templates. Figure 3-3 shows the host application performing a transaction in which a caller places an order. Figure 3-4 shows the host application creating a new account for a caller.

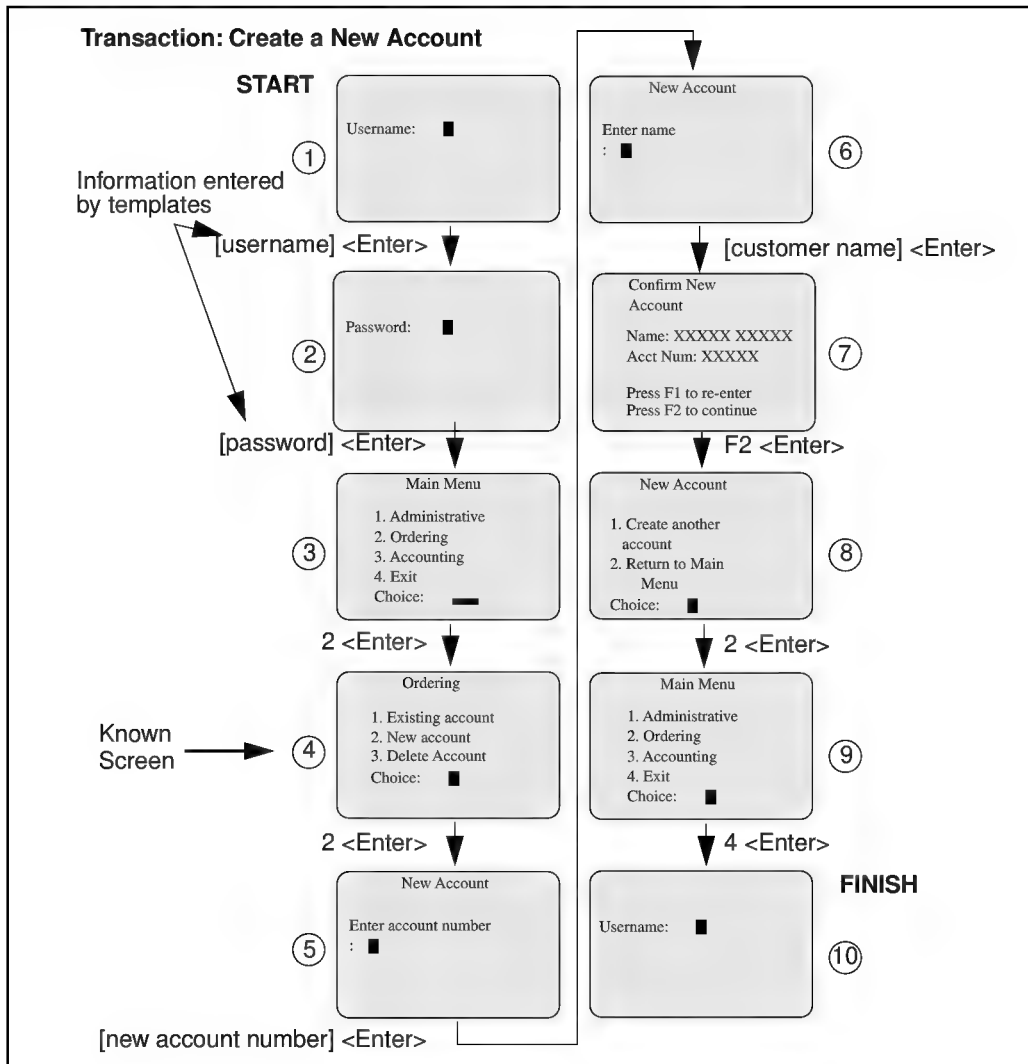
Note: The first step performed by the terminal operator is not performed by the Action template. It is performed by the Initial-Action template, described later in this chapter. The Initial-Action template handles the login and moves the application to the appropriate screen to begin the transaction.

Figure 3-3: Transaction: Placing an order using an existing account



3-8 Creating template files

Figure 3-4: Transaction: Placing an order using a new account



Note that the transactions shown in Figure 3-3 on page 3-7 and Figure 3-4 on page 3-8 share certain screens. The first four screens are the same for both transactions; therefore, a single action template can navigate through these common screens. In this example, the action template moves the transaction up to the Ordering screen (screen 4). This action template “parks” the transaction at screen 4, the screen from which different transactions start. This action template, referred to as the Initial-Action template, handles the beginning of the transaction.

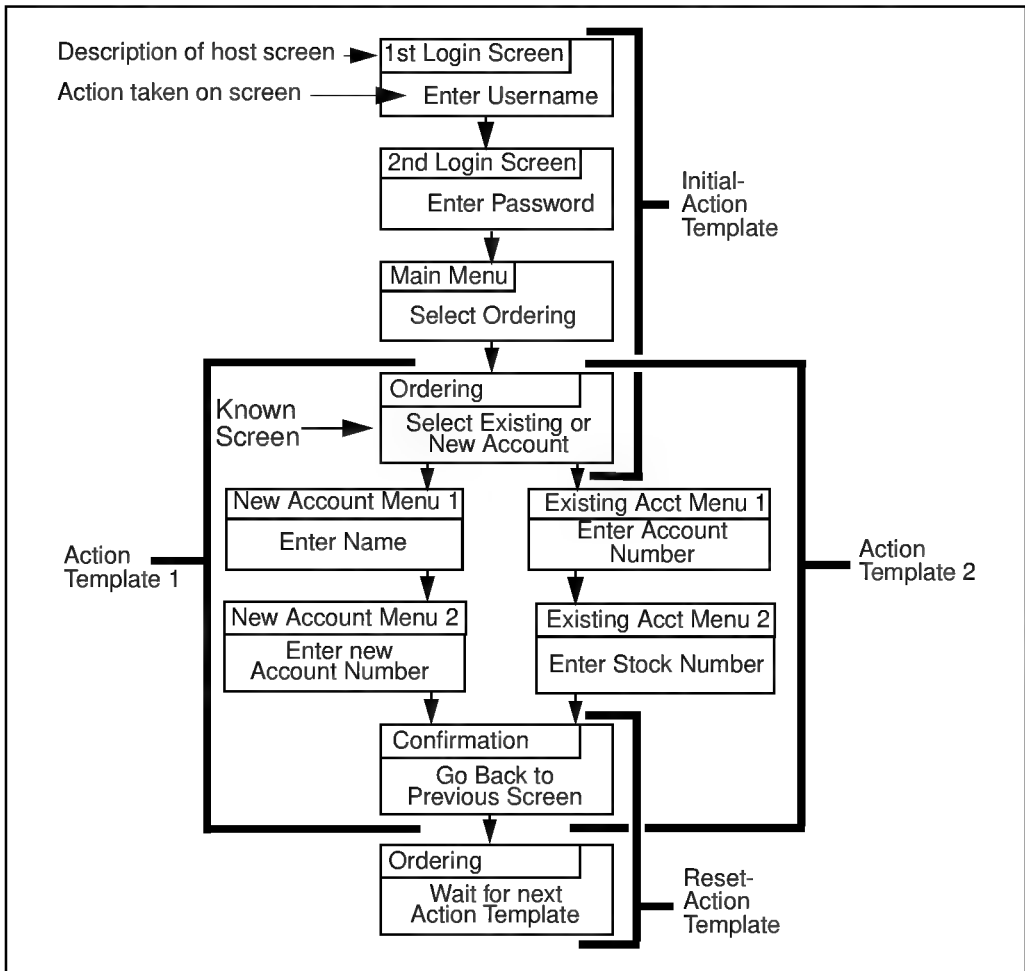
At screen 4, the transaction can branch to an “ordering” transaction or to a “new account” transaction. Therefore, a different action template takes over at this point, depending on which type of transaction your IVR Generator application is trying to accomplish. For this example, you create one action template to handle the “Ordering” transaction, and one action template to handle the “New Account” transaction. Note that one action template picks up on the screen where the previous action template (in this case, the Initial-Action template) leaves off.

The transactions once again merge at screen 8. At this point, another action template can take over and move the transaction back to the Main Menu (screen 9) so another transaction can begin. Since all action templates need to start from a specific screen (screen 9 or 3 in this example), you define an action template that moves the transaction back to the screen from which it started. This action template, called the Reset-Action template, handles resetting the transaction so another transaction can begin.

Occasionally, however, something may not function properly. A host application formatting inconsistency might make a line on the screen appear in a different location, or perhaps an unexpected screen appears. These types of problems affect action templates. To prepare for these situations, have a Logout-Action template that can return a failed transaction back to a point where the Initial-Action template can take over and attempt to repeat the transaction.

After navigating through the host application from a terminal, you may find it helpful to create a flowchart of the different screens you encountered. Figure 3-5 shows how a flowchart might look and demonstrates how these different types of action templates are suited to different segments of a transaction.

Figure 3-5: How action templates control different segments of a transaction



A flowchart of the host application provides a framework for creating action and screen templates.

Action Template syntax

Action templates are ASCII files created with a text editor. Your action templates must

- reside in /u/nortel/3270 or in a subdirectory below /u/nortel/3270

Note: If you make backups of your template files, do not store them in the /u/nortel /3270 directory, or in any subdirectory under /3270. TRS searches the /u/voice/3270 directory and any subdirectories within it for files with the .act or .scn extensions.

- assign the extension .act to the file name

For example, the path name for an action template called getbalance.act, is

/u/nortel/3270/getbalance.act.

All types of action templates follow the same syntax rules. The syntax of an action template is shown in Figure 3-6. The example is followed by an explanation of the syntax.

Figure 3-6: Sample action template

```
#comment
action_name app_name reset_action logout_action <manual look>
screen-template
screen-template
screen-template
:
:
:
.
```

#comment

The first line of the action template example is a comment. The comment line is not required, but it is recommended to describe the purpose of the action template. It is good practice to comment files so that you or others can easily make changes to the templates in the future.

Comment lines start with the “#” symbol and can be embedded anywhere in the action template. A comment takes the entire line; no non-comment fields may precede or follow the comment in that line.

action_name

The `action_name` is the action template file name without the `.act` extension. The `action_name` is required to begin the transaction. For example, if the file name of the action template is `getbalance.act`, then you enter `getbalance` for the `action_name`.

app_name

When you install the host link on your application server, create a `trs.conf` file that assigns TRS session numbers to the application on the host computer. The `trs.conf` file is described in Chapter 2, “Installing and configuring the host link software and hardware.” Choose a name for the host computer application; it does not need to match any actual application name on the host computer, but it must be used consistently throughout your action templates.

As an example, this guide uses the `app_name` “ord” to represent the Ordering application for the host computer.

The `app_name` you specify in the Action template must exist in the `trs.conf` file which is discussed in Chapter 2, “Installing and configuring the host link software and hardware.” IVR Generator passes the `app_name` to the host link, which uses the name to start the appropriate session with the host computer.

reset_action

In this field, enter the name of the Reset-Action template that moves the transaction back to the point from which the Action template started. Again, enter the name without the `.act` extension. For example, if the file name of your Reset-Action template is `reset.act`, then enter **reset** in this field.

The Reset-Action template is executed upon successful completion of the Action template.

logout-action

In this field, enter the name of the Logout-Action template that moves the transaction back to the point where the Initial-Action template can execute. Again, enter the name without the `.act` extension. For example, if the file name of your Logout-Action template is `logout.act`, enter `logout` in this field.

The Logout-Action template is executed if the Action or Reset-Action templates fail, or if the Action template fails and there is no Reset-Action template specified. If the transaction succeeds, the Logout-Action template is not executed.

After the host link uses the Logout-Action template to return a failed transaction to the initial screen, it executes the Initial-Action template after 30 seconds.

manual_mode (optional)

The `manual_mode` entry allows you to attach a session to a particular channel. You can then use the same session for consecutive IVR Generator transactions. This type of session is not released when the transaction is finished. To exit manual mode, you must execute a COMA cell in your IVR Generator application or process another Action template that does not contain the manual mode option. Chapter 4, “Using the COM cells to access the host computer,” describes how to use the COMA cell.

To specify manual mode, enter **manual** after the Logout-Action template field. If you omit `manual`, automatic mode is used for the template. In automatic mode, the session assigned to the Action template is free for use when the transaction is completed.

Note: You should not specify a Reset-Action template in an Action template that uses manual mode. Manual mode is designed to stay at a specific screen. The next transaction received by the session will start at the last screen of the Action template that used manual mode. This next transaction should use automatic mode and specify a Reset-Action template that brings the session back to the starting screen of the first Action template (that specified manual mode).

screen_template

In these fields, enter the names of the screen templates (screen templates are described in detail in later sections). Enter the name of the screen template without the filename extension. Enter the screen templates in the exact order they appear during the transaction. List each screen template on a separate line.

Initial-Action templates

The Initial-Action template serves two purposes:

- The template simplifies the development of host link applications by setting a starting point for the action templates.
- The template allows your applications to process host link requests more quickly since the transaction is waiting at a screen nearer the information screen. Therefore, the host link does not have to navigate through as many screens to retrieve information.

If your transactions do not start from the same screen, you may create multiple Initial-Action templates. Multiple Initial-Action templates can address a variety of different situations. You may have a situation where all of your transactions except one begin at the seventh screen of your host application. This one transaction begins at the second screen. Therefore, instead of creating one Initial-Action template that sets the starting point at screen 2 and forces all your transactions (except for one) to move through five additional preamble screens, you can create two Initial-Action templates. One Initial-Action template puts a certain number of terminal sessions at screen 2, and a certain number at screen 7. This places all transactions at the optimum starting point.

For example, if half of your transactions begin at screen 8, a quarter of them begin at screen 6, and a quarter begin at screen 2, then you define three Initial-Action templates. Each Initial-Action template places a portion of the terminal sessions at the appropriate screen.

Assign Initial-Action templates to terminal sessions in an ASCII file named `trs.conf`. This file automatically executes the appropriate Initial-Action templates at IVR Generator startup. See Chapter 2, “Installing and configuring the host link software and hardware,” for information on the configuration and syntax of `trs.conf`.

Sample Initial-Action template

The Initial-Action template follows the same format and syntax as defined for action templates in “Action Template syntax” on page 3-11. This action template works with the sample Initial-Action template in Figure 3-7.

Figure 3-7: Sample Initial-Action template

```
# Sample Initial-Action template. Filename:login.act
#
# action  app   reset      logout   manual
# name   name  action      action   mode
#-----
Login   Ord   GoToLogin FindLogin

# Screen
# Templates
#-----
EnterUName
EnterPswd
SelOrder
```


Action templates

Action templates are designed to start at the point where the Initial-Action template leaves off.

You must create a separate action template for each different transaction within the host application. Action templates are invoked by your IVR Generator application, so you need a one-to-one correspondence between your action templates and the different transactions your application tries to accomplish. The example shows two sample action templates and two different transactions (placing an order or creating a new account).

An action template must specify the same sequence of screens that you see as a terminal operator. Whenever the host link references an action template, it executes the screen templates listed in the action template, moving through the host application just as a terminal operator would navigate through screens to handle a transaction.

The example in Figure 3-8 illustrates an action template file which describes a transaction for placing an order.

Figure 3-8: Sample action template

```
# Sample Action template. Filename:PlaceOrd.act
#
# action   app   reset       logout    manual
# name    name  action      action    mode
#-----
PlaceOrd  Ord   GoToMenu FindLogin

# Screen
# Templates
#-----
EnterName
EnterAcct
Confirm
```

Reset-Action templates

Since action templates start where Initial-Action templates leave off, there must be a mechanism that returns the transaction to this starting point after the action template finishes. This mechanism is the Reset-Action template.

An action template leaves the transaction on the screen where the Reset-Action template starts, and returns the transaction to the screen where the action template started. In the same way that the Initial-Action template leaves the transaction at the point where the action template can start, the Reset-Action template starts at the point where the action template leaves off and returns to the action template. Just like the Initial-Action template, the Reset-Action template leaves the transaction at the point where another action template can start.

Depending on how the host application works, the Reset-Action template can return the transaction to the “Home Screen” in a variety of ways. The Reset-Action template returns to the “Home Screen” simply by “pressing” a single key, or it steps back through all the screens traversed by the action template. The Reset-Action template may combine these actions. The shortest consistent route to the “Home Screen” is the best strategy.

If you want to eliminate a Reset-Action template, enter a hyphen (-) in the `reset_action` entry in your action template. If the transaction succeeds and no Reset-Action template is specified, the host computer application remains parked at the screen where the transaction ended. If you do not specify a Reset-Action template and the transaction fails, the Logout-Action template (described later in this chapter) is executed.

When you create a Reset-Action template, do not specify Reset-Action or Logout-Action templates within it. Figure 3-9 shows a sample Reset-Action template.

Sample Reset-Action template

Figure 3-9 illustrates the Reset-Action Template that is specified in the sample action template in Figure 3-8.

Figure 3-9: Sample Reset-Action template

```
# Sample Reset-Action template. Filename:GoToMenu.act
#
# action app reset logout manual
# name name action action mode
#-----
GoToMenu Ord - -
# Screen
# Templates
#-----
ExitConfirm
ExitAcct
```

Logout-Action template

The Logout-Action template only executes if one of the action templates fails, which occurs when one of the screen templates specified in the action templates did not navigate through its screen.

The Logout-Action template moves the transaction from an unknown state to a known state. The Logout-Action template determines the progress of the transactions. When the Logout-Action template is invoked, it begins executing a series of screen templates. When one of the screen templates “hits” (meaning it had an effect on the host screen), the sequence of the screen templates is such that it begins moving the transaction back to a logout. But when host errors occur, the sequence of host screen may be incorrect; there is no guarantee that a “hit” on one screen automatically puts your transaction on the path to a logout.

The Logout-Action template executes a sequence of screen templates. This sequence of screen templates is executed in order, regardless of the success or failure of each screen template. If the final screen template in the sequence successfully executes, the Logout-Action template considers the logout complete and invokes the Initial-Action template (after 30 seconds). If the final screen template in the sequence does not successfully execute, the sequence repeats.

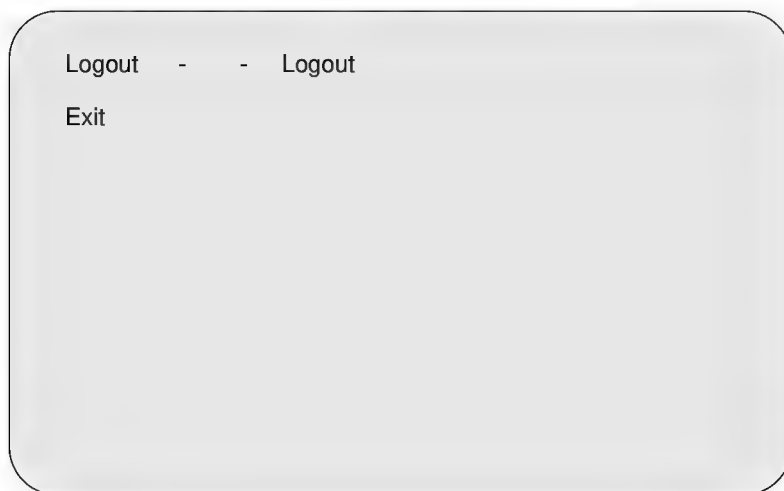
Create a successful Logout-Action template by placing screen templates in a sequence that enables the best movement for the transaction to a logout. For example, every host screen should contain a certain key that always exits or places you on a known screen. Then your Logout-Action template would only have to specify a single screen template—one that presses that key. However, not every screen has an “Exit” key. Perhaps half the host screens have “Exit” keys and the other half have “Next” and “Previous” keys. Here, the best strategy for the Logout-Action template would be to loop between those that press the “Previous” key and screen templates that press the “Exit” key.

Sample Logout-Action template

Entering a hyphen (-) for the logout-action entry indicates that a Logout-Action template is not specified. If neither a Reset-Action template nor a Logout-Action template is specified and the transaction fails, then the host computer application remains at the point where the transaction failed. Subsequent transactions that attempt to use this session also experience errors because the screen where the session remained would not be the expected starting screen unless the transaction can start from any screen.

When you create a Logout-Action template, do not specify Reset-Action or Logout-Action templates. Figure 3-10 shows a sample Logout-Action template for Figure 3-8.

Figure 3-10: Sample Logout-Action template



Heartbeat Action template

You specify the heartbeat in this format: `actiontemplate@n`, where *n* is the number of seconds between each execution of the action template when the session is idle.

The Heartbeat Action template uses the same syntax as the action templates described earlier in this chapter, and usually only specifies a single screen template. That screen template is usually the last screen specified in the Initial-Action template. Typically, your screen template only includes a key-descriptor line, usually the **<Enter>** key and validation tag.

Screen templates

A screen template tells the host link exactly how to maneuver through one specific host screen. A screen template can perform several actions on the host screen as it maneuvers through the screen. These actions include

- locating “landmarks” on the screen to verify that the screen template is on the correct screen
- locating information to return to your IVR Generator application
- sending information from your IVR Generator application to the host application
- entering keystrokes and navigational commands just as a terminal operator would enter these keystrokes and commands

By using an action template to specify the order of screen templates, you provide the host link with all the information it needs to navigate through a transaction from start to finish. See Figure 3-2 which illustrates the relationship between action and screen templates.

Define one screen template for each host screen. Each host screen contains fields. A field is an area on the screen where text appears. A field may be writable, meaning the host application expects information to be entered at this point. A field may be a region where the host application displays data, or it may be a location where static characters are displayed. For example, the header on the screen is considered a field.

A screen template can use a field on the host screen for several purposes:

- as a reference point or navigational aid that allows the screen template to verify that it is on the correct screen
- as a target for information sent from your IVR Generator application
- as a source of information to return to your IVR Generator application

A transaction consists of entering specific data in certain fields, retrieving data from other fields, and pressing keystrokes. The screen template locates all the fields on the screen that the host link uses to process a transaction. Include only the fields that are used in the transaction in the template.

You give the screen template the location of the fields on the host screen by specifying them in “row, column” format in the field descriptors.

For example, a screen template may perform the following steps in order to maneuver through the login screen shown in Figure 3-11.

Figure 3-11: Fields on the host screen



- 1 Each screen template first validates that it is on the correct screen. Therefore, ensure that each screen template checks for a specific piece of text on the host screen. This piece of text is known as a validation tag and should uniquely identify this screen from *all* other host screens.

If host screens do not contain a unique identifier, use the tag that is the most unique. Always define a validation tag.

In Figure 3-11, the word “Username” provides the validation tag since it only appears on this particular screen.

You specify the location of the validation tag in “row, column” format. Locate the first character of the validation tag. From the upper left corner of the screen, count down the number of rows, then over the number of columns to this character.



In this example, the row and column location of the validation tag corresponds to the 'U' in 'Username.' Specify 4,1 for the location of the validation tag.

- 2 The screen template then locates the first space in the writable field. As with locating the validation tag, the coordinates you enter represent the first space in the field. The following picture illustrates how to determine the coordinates of a text entry field:

Note: When determining the coordinates of a text entry field, take into account any punctuation and space after the prompt.

11

4



In the preceding illustration, the cursor after the Username prompt represents the first character space in the field. This is the location you want to tell your screen template to use to enter information. The coordinates of the cursor are 4,11. This information comprises the first part of the field descriptor.

- 3 The screen template enters the text **mainuser** in this field. Once the screen template locates the text entry field, it transmits characters into the field. These characters can come from your IVR Generator application, or you can code the information directly into your screen template. These characters comprise the second part of the field descriptor.

In your actual screen template, you combine steps 2 and 3 in the field descriptor.

At this point, your screen template can return information to your IVR Generator application similar to the same way data is transmitted to the host application. This procedure is described in detail in “Screen template syntax” on page 3-28.

- 4 The screen template presses the <Enter> key. Once the screen template enters its information in the text entry field or reads its information, it indicates that a key needs to be pressed in order to tell the host application to move to the next screen. This portion of the screen template is known as the key descriptor. The key descriptor does not need associated location coordinates.
- 5 The screen template waits for a specified length of time for the next screen to display. Sometimes the host does not display the next screen instantly. This portion of the screen template is known as the sleep descriptor.

The exact syntax of each of these descriptors is outlined in the following section.

Screen template syntax

Screen templates are ASCII files you create with a text editor. Your screen templates must

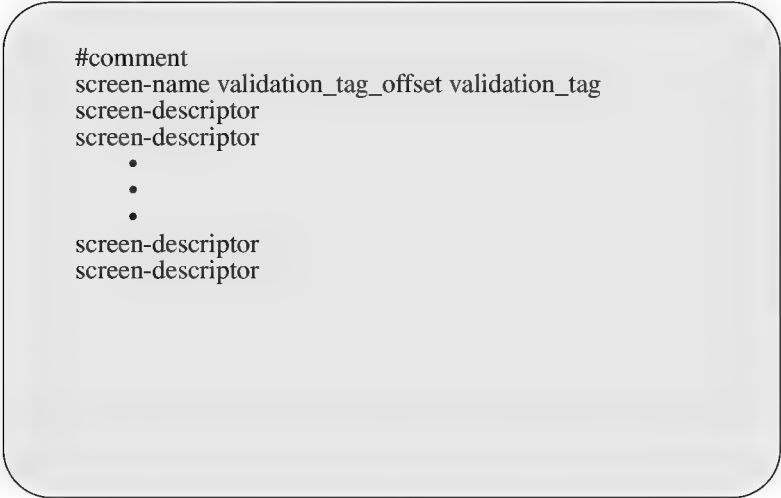
- reside in `/u/nortel/3270` or in a subdirectory below `/u/nortel/3270`
- use the filename extension `.scn`

For example, if you created a screen template called `EnterName.scn`, it uses the following path:

`/u/nortel/3270/EnterName.scn`

The syntax of a screen template is shown in Figure 3-12.

Figure 3-12: Screen template syntax



```
#comment
screen-name validation_tag_offset validation_tag
screen-descriptor
screen-descriptor
.
.
.
screen-descriptor
screen-descriptor
```

An explanation of each entry in the screen template follows.

#comment

Comment lines start with the “#” symbol which you can embed anywhere in the screen template. A comment takes the entire line; non-comment fields cannot precede or follow the comment on that line.

screen-name

The first non-comment line specifies the name of the screen template. This is the screen template file name without the .scn extension.

validation_tag_offset

This entry specifies the row and column of the validation tag. There are two additional special uses of the validation tag offset, 0,0:

- You can enter 0,0 for the validation tag and a hyphen (-) as the validation tag to validate the screen. For example, to execute a command at the system prompt, if you ensure the screen contains a system prompt, you do not need to verify your location at a specific screen.

Note: You should validate screens whenever possible. This ensures that data for the host computer application is sent to the correct screen and data sent back to the IVR Generator application is from the correct screen.

- You can also enter 0,0 to tell the host link to search the screen for the validation tag if you do not know the exact location of the validation tag, or if the location of the validation varies. Enter 0,0 for the offset, and then enter the validation tag in the appropriate field.

validation_tag

The validation_tag is an identifier on the host screen that uniquely identifies the screen to the host link. It allows the screen template to make sure it is on the correct screen. If the screen template cannot find its validation tag, the screen template's governing action template executes the Logout-Action template.

The text entry appears in the same location every time and is unique to that screen. For example, "Username" always displays at location 4,1 whenever the login screen is displayed.

If the screen you need to validate is a blank screen, enter the following line for the items: screen-name, validation tag offset, and validation tag.

```
blank 1,1 BLANKSCREEN
```

The word "blank" is a placeholder; the 1,1 and BLANKSCREEN are required fields.

screen-descriptor

In order for the host link to enter information into or retrieve information from a field, the host link requires two pieces of information:

- the location of the field
- a description of the action to perform on the field

You provide this information in the screen descriptor. You can define multiple screen descriptors in one screen template, but you should use one screen template for each host application screen. Figure 3-13 shows the syntax for the three types of screen descriptors: field descriptors, key descriptors, and sleep descriptors.

Figure 3-13: Screen-descriptor syntax

```
# Field descriptor
position field-name field-action

# Key descriptor
position > keyname

#Sleep descriptor
position @ sleep time
```

Note: The order of the field-descriptor lines in your screen template file determines how data is passed to the host by the COM cell and retrieved from the host by the COMO cell. Chapter 4, “Using the COM cells to access the host computer,” describes how to use IVR Generator cells to retrieve data from the host computer.

position

This entry indicates the location of the field on a terminal screen. Enter the exact location of the field in row, column format.

If you do not know the exact location of the field from which TRS is reading, or if the location of the field varies, you can enter 0,0 and TRS searches for a match to the name specified in field-name. For unformatted, character-based applications, you must specify the exact location.

In SNA transactions, writing to the terminal screen occurs at the current cursor position. Therefore, if the field I/O action is writing text to the host screen, TRS writes the text to the screen at the current cursor position regardless of the row and column you specify.

If you enter 0, 0 and a hyphen (-) for the row, column, and field-name entries, the field I/O specified is executed at the current cursor location.

The position for key descriptors and sleep descriptors will always be 0,0.

field-name

This entry identifies the field the host link accesses on the terminal screen. If the field-descriptor is 0,0, the field-name must uniquely match the text on the screen or must be a hyphen to indicate that the field is at the current cursor position. When entering the field name, the host link right-justifies all field names and removes all extra white space. If the transaction fails, this field identifies the current screen so that the Reset-Action template is executed.

If you enter "0,0" and a hyphen (-) for the row, column and field-name entries, the field I/O specified is executed at the current cursor location.

To specify a location on the screen that does not have an associated field-name, use a hyphen (-) for the field-name entry. You must specify a location, such as 24,8, for the location of the action specified by the field-I/O entry.

field action

This entry indicates the action to be taken on the field. Table 3-1 explains the valid entries for this field.

Table 3-1: Special symbols used in screen descriptors

Entry	Description
*	Transfers the contents of the next IVR Generator input buffer to the indicated field.
\$num	Transfers the contents of the field into the next IVR Generator output buffer. The number portion of the entry num is a number in the range 1 to 31 and represents the number of characters returned from the host. If you do not assign num a value, the host link places 31 characters into the buffer.
text	Places a text string in the field. If any of the special characters listed in this table are included in the text, enclose the entire text in quotes (for example, "\$abc").
BLANKOUT	Clears host link memory space associated with the host application screen before retrieving output.
“ ”	If text contains an asterisk or dollar sign, enclose them in quotation marks.
%	The following text is an internal variable. Internal variables are used within the screen templates only, and are not passed to the host application.
>	Indicates a key descriptor; a keyname follows.
@	Indicates a sleep descriptor; a sleep time follows.

Note: Place the asterisk and dollar sign characters in quotation marks if your field-descriptors require their use without their associated buffer commands.

When entering a sleep-descriptor, separate the entries on the line by white space or tab characters. If you enter text for the field action, the host link ignores any white space you include with the value until it encounters the new line character (for example, when the value includes several words).

Note: If the application running on the host is stream-based, meaning the screen scrolls as the user enters data and retrieves responses, enter the field-descriptor as follows:

0,0 - BLANKOUT

In this instance, the field-descriptor clears host link memory space associated with the application screen so that your application call flow knows where to retrieve the appropriate data.

Searching a screen with the field descriptor

If you need to enter or retrieve information from a field that does not appear in a consistent location on the screen, the field descriptor can search for the field. To do this, enter the following:

- **0,0** for the location
- word or characters located immediately to the left of the writable field
- field action command

The screen template searches the screen for the word you specify and then performs the action command in the next writable field.

keyname

This entry shows the name of the key for this screen. Table 3-2 lists valid keys.

Table 3-2: Valid function keys

Key names		
ATTENTION	FORWARDWORD	PF11
BACKSPACE	HOME	PF12
BACKTAB	INSERTCHAR	PF13
BACKWORD	NEWLINE	PF14
CLEAR	PA1	PF15
CURSORDOWN	PA2	PF16
CURSORLEFT	PA3	PF17

3-34 Creating template files

CURSORLEFTDBL	PF1	PF18
CURSORRIGHT	PF2	PF19
CURSORRIGHTDBL	PF3	PF20
CURSORUP	PF4	PF21
DELCHAR	PF5	PF22
DUP	PF6	PF23
ENTER	PF7	PF24
ERASEEOF	PF8	RESET
ERASEINPUT	PF9	SYSREQUEST
FIELDMARK	PF10	TAB

sleep-time

Specifies the number of seconds that the session should sleep. For example, to wait 25 seconds, you would code the sleep descriptor as follows:

0,0 @ 25

Sample screen template

The example in Figure 3-14 illustrates a screen template file that obtains the balance from the screen shown in Figure 3-12.

Figure 3-14: Screen template example

```
#Screen template file to obtain the current balance;  
Customer 1,1 Account  
#places balance into buffer  
  
0,0 Balance: $  
  
0,0 > ENTER  
  
0,0 @ 5
```

In Figure 3-14, the second line describes the screen template file. The screen-name is `Customer`, the name of the screen template file without the `.scn` extension. The `1,1` represents the location of the validation tag on the screen. The row is listed first, followed by the column. `Account` is the screen validation tag.

The third line is a comment that describes what to do with the balance. The fourth line is the field-descriptor that describes an action to take. This field descriptor is going to find an exact match to `Balance:` and place the contents of the field into a buffer. Depending on what you want to do with a field, the field-descriptor line varies. For example, you can enter the last line in the example as

```
2, 48      -      $
```

This example places the contents of the field starting at 2,48 into the next buffer.

Testing your action and screen templates

When you finish creating your action and screen templates, test them to ensure that the syntax is correct.

Test your action and screen templates with `trs -c`.

Using trs -c

Once you have created your action and screen templates and placed them in `/u/nortel/3270`, do the following:

- 1 Change the directory. Enter `cd /u/nortel/exe`
- 2 Type `./trs -c`

You should see output similar to the following:

```
TRS Checking Tools.....Version 1.00.00
    TRS was compiled for.....SCO
    Total Channel Allocation..12
    Total screen templates....23
    Total action templates....14
    SNA Based Product.....System Strategies
    Communication Board(s)....1
    Total defined sessions....16
    Session mapping.....No
    The Screen Templates.....OK
    The Action Templates.....OK
    appl name:[ssi]
    BD:0 SS:002-017 protocol: SNA
    Init action[ssiInit],idle_action[]
    Runtime configuration.....OK
```

If an action or screen template contains a syntax problem, `trs -c` indicates which template has the problem and specifies the problem. The `trs -c` procedure also checks for the presence and syntax of the additional data files required by the host link.

Checking the trs.conf file with trs -c

If the `trs.conf` file is not created or does not reside in the directory `/u/nortel/3270`, then the output from `trs -c` is as follows:

```
TRS Checking Tools.....Version 1.00.00
trs.conf file not defined
```

If `trs.conf` contains a syntax error, the output from `trs -c` contains:

```
ERR: Unable to load configuration file
```

If `trs.conf` cannot access the specified Initial-Action template, a syntax error registers and you get this message. See Chapter 2, “Installing and configuring the host link software and hardware,” for more information on `trs.conf` file.

Checking the `trs.sdf` file with `trs -c`

If you did not create the `trs.sdf` file or it does not reside in `/u/nortel/3270`, the output from `trs -c` contains:

```
trs.sdf file not defined
Running as a Dummy Process
```

See Chapter 2, “Installing and configuring the host link software and hardware,” for more information on `trs.sdf`.

Checking screen templates with `trs -c`

If there is a syntax error within a screen template, the output from `trs -c` contains lines similar to the following:

```
ERR: No header data for Screen file
../3270/clear.scn
Running as a Dummy Process
ERR: Parse string xxxxx of screen clear
Running as a Dummy Process
```

The text after the `ERR:` describes the error and then indicates which screen template contains the error. In these lines, the screen template containing the error is `clear.scn`. The first error message indicates that there is a syntax error in the header of the screen template `clear.scn`. The second error message indicates that `trs -c` encountered a problem trying to parse the string `xxxxx` in the screen template `clear.scn`.

Checking action templates with `trs -c`

If an action template contains a syntax error, the output from `trs -c` contains lines similar to the following:

```
ERR: read head data from action file
```

```
../3270/exit.act  
Running as a Dummy Process  
ERR: Screen clear of Action exit not found  
Running as a Dummy Process
```

The text after the ERR: describes the error, then indicates which action template contains the error. The first error message indicates that there is a syntax error in the header of the action template `exit.act`. The `trs -c` also checks to make sure that the screen templates listed in the action template actually exist in the `/u/nortel/3270` directory. The second error message indicates that `trs -c` could not find the screen template `clear.scn` listed in the action template `exit.act`.

TRS log

This section provides information on how to debug SNA templates using the TRS log. The TRS does not create a log unless the `startup` file is modified to put the TRS in verbose mode.



Attention

Run your system in verbose mode for diagnostic purposes only. Set a time limit and monitor your system memory using the `sar` command.

To put the TRS in verbose mode, modify the `/u/nortel/startup` file. On the `trs` line, add the `-v` flag as follows:

```
./trs -b -v
```

Restart IVR Generator and SNA to create the log. The log is stored in the `/u/nortel /3270` directory as the `trs.log`. This log shows all data sent to the host and all data received from the host.

The TRS log assists in the development of templates or when debugging a problem. The log varies from customer to customer because most of what is used to search through the log consists of parts of the host screens. For example, a company has its customers' phone number on each screen, so you search on that to find transactions.

All IVR Generator calls go by channel number but the TRS log goes by LU number. At the beginning of a transaction, lines in the log indicate the channel and LU numbers. Use these lines to determine what LU the transaction is using, for example,

```
14:09:35: CH=001 BD0SS005  
Parse:Action_buffer:Billing
```

This example shows that channel 1 is conducting a transaction on LU 5. If the channel number is 999, then no channel maintains control of the LU. When pings are issued, they are conducted with a channel of 999 because it is the TRS conducting the ping, not a caller.

To find a transaction, you can conduct a variety of searches. By using the name of the action template without the .act extension, you find the start of the transaction. Searching for the number of the channel you were on in the format CH=xxx finds the start of the transaction, too. Searching for a string that was sent in the COMI finds the buffers being sent in to the transaction. Searching for a part of a host screen can find the body of that screen.

Once a part of a transaction is found, the transaction can be followed step by step until its end. First, find the LU number of the transaction. It is in the format of BD0SS005 where the number on the end is the LU number. By searching on this, you can find most of the lines that are part of the transaction.

As you step through the transaction, you see different actions occurring. Here are some of these actions in order.

The following example shows the action template that was called from the COMI:

```
14:09:35: CH=001 BD0SS005
Parse:Action_buffer:Billing
```

The following example shows how the host screen appears. In this case, it is blank:

```
CH=001 BD0SS005 * Read Input Screen BODY BEGIN *
```

(Blank Screen)

The following example shows the name of the screen template.

```
14:09:35: Expected screen is: sCustomerInfo
```

The following example shows the valid tag in which the screen template is looking for input to screen:

```
14:09:35: Validate_scn: Valid Tag is BLANKSCREEN
14:09:35: Checking the BLANK screen
```


The following example shows strings sent from the COMI; the screen template is sending them to the screen:

```
14:09:35: BD0SS005 STR=Bill
14:09:35: BD0SS005 STR=J
14:09:35: BD0SS005 STR=Smith
```

The following example shows the strings that are written to the screen:

```
14:09:35: Writing Bill (len= 4) to P.S 1
14:09:35: SNA: BD0SS005 Copy String to PS Succeeded
14:09:35: Writing J (len= 1) to P.S 5
14:09:35: SNA: BD0SS005 Copy String to PS Succeeded
14:09:35: Writing Smith (len= 5) to P.S 10
14:09:35: SNA: BD0SS005 Copy String to PS Succeeded
```

In the following example, an Enter key is being sent. Once the Enter is sent, wait for the next screen to return. To find the next action, search for an LU number like BD0SS005.

Note: Never send a string to the screen that is empty. A string copy failure occurs if the len = 0.

```
14:09:35: BD0SS005 <==Write Screen Done
14:09:35: BD0SS005 <==Write Screen to the Host is
done **
Username: Bill J Smith
14:09:35: BD0SS005 <==Send with AID key ENTER
```

The following example shows the next screen that came from the host:

```
CH=001 BD0SS005 * Read Update Screen BODY BEGIN *
User Bill J Smith
Account Balance: $100
Amount Due: $25
Due Date: 120197
```

The following example shows the information retrieved from the screen per the request of a screen template. This data is placed in the COMO buffers.

```
14:09:37: Hold: CH=001 Reply Code=0 with 8 buffers
BUF #0 =25
BUF #1 =120197
```

Chapter 4: Using the COM cells to access the host computer

This chapter explains how to use the COM cells to develop an IVR Generator host communications application that processes one or more terminal sessions. Specifically, this chapter contains the following sections:

- “Setting the COMI cell parameters” on page 4-2
- “Setting the COMO cell parameters” on page 4-5
- “Using the COMA cell” on page 4-8
- “Determining successful COM cell communication” on page 4-9
- “A sample application using the COM cells” on page 4-10

Begin the application development process as follows:

- Determine the telephone interaction with the caller and create the corresponding IVR Generator call flow.
- At the point in the call flow where you require interaction with the host computer, insert a COMI cell.
- Insert a COMO cell to receive the output from the IBM host.

Note: You must always follow a COMI cell with a COMO cell, even if you do not require the return of any data. Connect Error branches to a COMA cell to resolve any potential problems.

Setting the COMI cell parameters

The COMI cell starts a transaction with the host. This cell invokes an action template which contains the commands needed for the communication transaction.

COMI cell parameters page

Figure 4-1 shows the COMI cell parameters window. This section explains how to use this window.

Figure 4-1: COMI cell parameters window

Cell 42 **COMI** input to Host

Start Getbalance Transaction

Comments

Call Audit Enabled? ☒ Yes ☐ No

Call Audit Information: 212

Action Template: getbalance02

More Input? ☒ Yes ☐ No

75

Timeout (seconds)

Input Buffers

Buffer Count: 1

1. ACCOUNTNUMBER

2.

3.

4.

Apply Cancel Help

The following sections describe what you should enter for each COMI cell parameter field.

Cell name

The cell name should identify the function of the cell. For example, the cell name shown in Figure 4-1 is “Start Getbalance Transaction.”

Comments

In this space, enter any information about the COMI cell.

Action template

Enter the name of the “action template” in quotes (leaving off the .act suffix). Figure 4-1 shows “Getbalance” as the name.

More Input?

If you need to send more than 10 input buffers to the host computer, click the “Yes” button; if you are sending 10 or fewer input buffers, click the “No” button. If you specify “Yes,” place additional COMI cells on the drawing board until you have enough input buffers. Connect the “SUCCESS” branch of the first COMI cell to the “INPUT” branch of the following COMI cell, and then connect the rest of the COMI cells in this manner. The final COMI cell should have the “No” button selected for the “More Input?” parameter.

Timeout

Select the number of seconds you want the TRS process to wait for the transaction to complete by moving the slider under “Timeout.” You can move the slider from 1 to 75 seconds. This indicates the maximum time for the COMI cell to start the transaction and send the necessary input buffers, and for the COMO cell to receive the output buffers. The TRS process rejects the transaction and sends back a timeout if the action is not performed within the time allotted.

Note: The application takes the timeout branch of the COMO cell if a timeout occurs.

Buffer count

Leave this cell blank as IVR Generator automatically calculates the number of buffers when you click the “Apply” button.

Input buffers

Enter the names of the input buffers containing the information to pass to the TRS process. For example, the input buffer shown in Figure 4-1 is ACCOUNTNUMBER (without any quotes). In this example, you program your application to place the customer's account number in the ACCOUNTNUMBER buffer before the COMI cell executes. ACCOUNTNUMBER is a user-defined buffer; you can use any name for your applications, or you can use system buffers.

Note: Enter the input buffers in the same order that the host computer uses them.

If you need to use more than 10 input buffers in your application, make sure you select "Yes" for "More Input?"; then, string COMI cells together to create input buffers. Connect each additional COMI cell to the previous cell's "SUCCESS" branch.

Setting the COMO cell parameters

Once you set the COMI cell to send information, then code the COMO cell to receive information. You must place a COMO cell directly after the COMI cell to complete the transaction, even if the host computer is not sending any data to any output buffers. The TRS process verifies that the transaction completed successfully or that there was an error in the COMO cell.

The COMO cell parameters page

Figure 4-2 shows the COMO cell parameters window. This section explains how to use this cell.

Figure 4-2: COMO cell parameter window

The screenshot shows a window titled "COMO Parameters". Inside, there's a "Cell ID" field with "COMO" and "Output from Host" with "Receive Balance". Below that is a "Comments" field. Then "Cell Asset Enabled?" with "Yes" selected and "No" as an option. "Cell Asset Information" is set to "00000000". "Blocking?" has "Yes" selected and "No" as an option. The "Output Buffers" section shows "Buffer Count" as "1" and a list of buffers with "BALANCE" in the first one. At the bottom are "Apply", "Cancel", and "Help" buttons.

The following sections describe what you should enter for each COMO parameter field.

Cell name

The cell name “Receive Balance,” shown as an example in Figure 4-2, identifies the function of the cell.

Comments

In this space, enter any information about the COMO cell.

Blocking?

Activating blocking tells the COMO cell to wait for the transaction to complete before continuing to the next cell. The application waits until the TRS interaction ends before it continues through the IVR Generator application (following the “END OF DATA” or “MORE DATA” branch; only one of these cells should have a branch). If you select “No,” the COMO cell receives a status code from the TRS process. Refer to the “COMO cell branches” section that follows.

Buffer Count

Leave this field blank as IVR Generator automatically calculates the number of buffers when you click on the “Apply” button.

Output buffers

Enter the name of the buffers that accept the output from the host computer. Figure 4-2 shows BALANCE (no quotes) as the only output buffer for this example.

Note: Enter the output buffers in the same order that the TRS process uses them.

If your application uses more than 10 output buffers, string COMO cells together until you have enough buffers. Connect the input branch of each subsequent COMO cell to the “MORE DATA” branch of the previous COMO cell.

COMO cell branches

As shown in Figure 4-3, the COMO cell contains several branches.

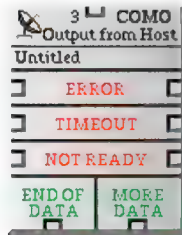
Figure 4-3: COMO cell

Table 4-1 describes each of the COMO cell branches.

Table 4-1: COMO cell branches

Branch	Reason Taken
ERROR	If there is any problem with any part, the host transaction will cause the application to take the error branch.
TIMEOUT	The transaction took more than its allowed time as specified in the COMI cell.
NOT READY	The transaction did not complete and Blocking was set to "No."
END OF DATA	The transaction successfully completed.
MORE DATA	The transaction requires additional output buffers.

For each COMO cell, use either the "END OF DATA" branch or the "MORE DATA" branch, but not both. If you use the "MORE DATA" branch, the next cell must be another COMO cell. The last COMO cell should use the "END OF DATA" branch.

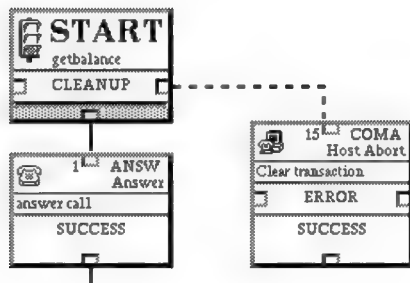
Using the COMA cell

Use the COMA cell to abort a transaction in progress and release the session if you selected manual mode in the application template. For example, you can use a COMA cell in the cleanup branch of the START cell to deal with an error such as a caller hanging up in the middle of a transaction.

COMA cell parameters window branches

The COMA cell frees all memory and buffers associated with the COMI and COMO cells. Figure 4-4 provides an example of a COMA Cell Window. The cell parameters contain a cell name field and a comments field. In this example, the COMA cell has “Clear transaction” as its cell name.

Figure 4-4: COMA cell in the cleanup handler



Determining successful COM cell communication

A System Activity Monitor (SAM) trace returns specific codes that allow you to determine if the cells communicated with the host. Table 4-2 lists these codes.

Table 4-2: COMI and COMO reply codes

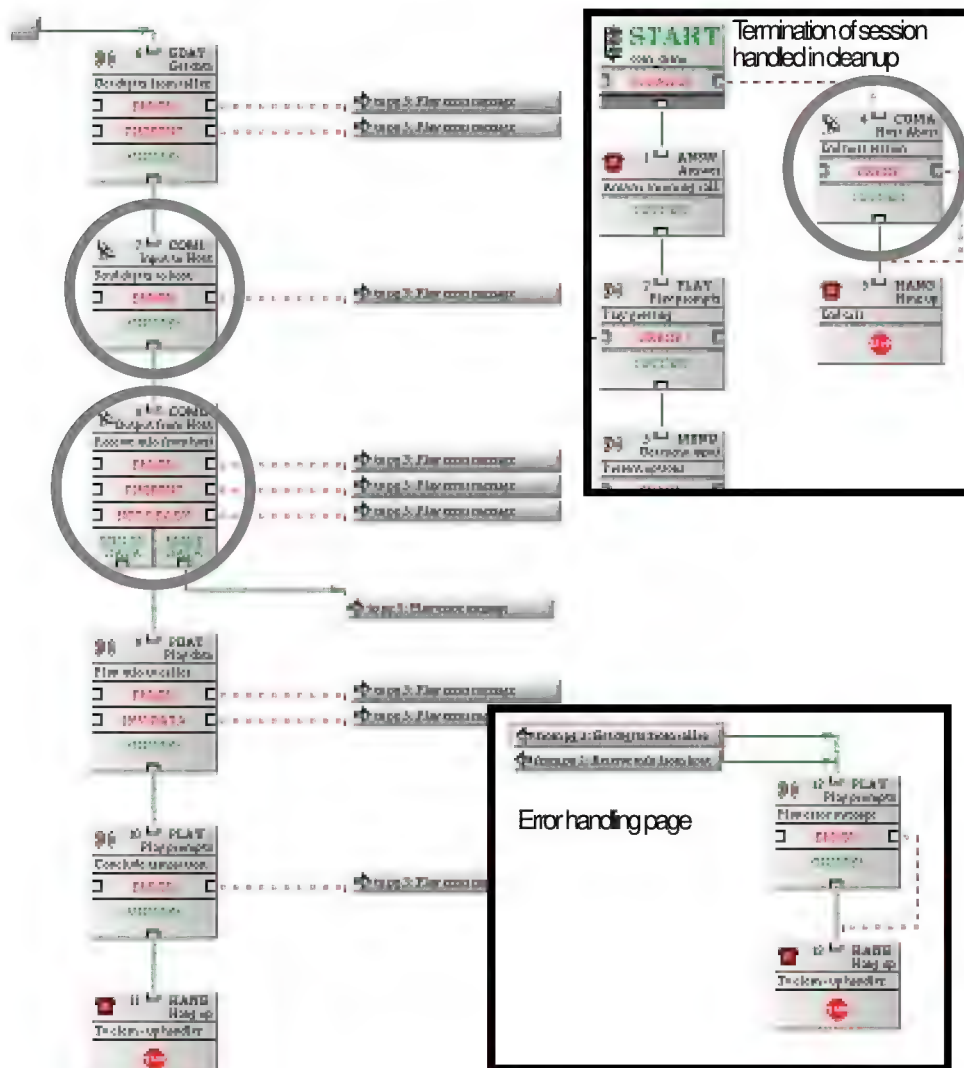
Return code	Description
0	Completed transaction; returned up to 10 buffers to the IVR Generator application.
1	The transaction failed; memory resource error or a command sequence failure.
2	The transaction failed; process failure.
3	The transaction failed; process timeout.
4	Not used.
5	The transaction received a "Not Ready" response from the host link.
6-7	Not used.
8	Transaction succeeded; more input required.
9	Transaction succeeded; more output required.

A sample application using the COM cells

Figure 4-5 shows a sample IVR Generator application that uses the COMI, COMO, and COMA cells to retrieve a customer's account balance stored on a host computer. This application uses the action and screen templates created in Chapter 3, "Creating template files." A caller activates this application by pressing "1" after hearing the prompt played by the MENU cell. A separate action template and its associated screen templates need to be created to support menu choice 2.

To illustrate the logic of the call flow, these applications are shown as one-page applications with no error branches. These applications span several pages and all error branches connected appropriately.

Figure 4-5: IVR Generator application accessing the TRS process from the COMI and COMO cells



Just before this application executes, the Initial-Action template is executed (when IVR Generator is started on the application server). Once a call from a customer is received, the IVR Generator application performs the steps listed in Table 4-3.

Table 4-3: Sample transaction processing steps

Cell 1	The ANSW cell answers the incoming call.
Cell 2	The PLAY cell plays greeting and introductory prompts to the caller.
Cell 3	The MENU cell plays a prompt instructing the caller to press “1” to get an account balance, or press “2” to obtain an account update. In this example, the caller presses “2” for an account update.
Cell 4	The COMA cell is executed after the caller hangs up. This terminates the host session.
Cell 5	The call ends.
Cell 6	When the caller presses “2” on the telephone, the GDAT cell plays the prompt requesting the caller to enter an account number. The number is stored in the ACCOUNTNUMBER buffer.
Cell 7	The COMI cell starts the transaction. The action template specified in the COMI cell parameter window is “Getbalance,” the same template used in the example transaction described in Chapter 3, “Creating template files.” The COMI cell provides the contents of the ACCOUNTNUMBER buffer to the host computer application.
Cell 8	The COMO cell receives the customer’s balance from the BALANCE buffer.
Cell 9	The PDAT cell plays the contents of the BALANCE buffer to the caller.
Cell 10	The PLAY cell plays prompts to conclude the call.
Cell 11	The caller hangs up and the application moves to the cleanup handler (cells 4 and 5).
Cell 12	In the event of an error in the main call flow, this PLAY cell informs the caller that an error has occurred.
Cell 13	The call moves to the cleanup handler (cells 4 and 5).

If the caller hangs up before the transaction is completed, the COMA cell in the cleanup branch of the START cell clears the memory and buffers associated with the application.

Chapter 5: Troubleshooting

This chapter discusses the tools available for gathering and viewing debugging information, and lists host link error messages.

Debugging the EXPRESS application

EXPRESS provides three sets of tools that gather and view information for debugging the EXPRESS application:

- Log file
- Trace file
- information from Verbose processes

The Log file and Trace file components are integrated into both the character-based user interface and graphical-based user interface. Verbose processes information is initiated by modifying the `express_adm` script.

The EXPRESS product also provides information on the status of the connection between host and application server.

Log file

The Log File records events, errors, and abnormal conditions in an ASCII file. Multiple ASCII files (segments) comprise a log. The user can define the maximum number of events that the file can contain. When the maximum number of events is reached, the existing file is closed and a new file is started.

The type of status can include all activations and deactivations, detected recoverable conditions (XID or INACTIVITY timeouts), and any nonrecoverable conditions (sense code information) detected by the background processes.

This file provides the basic functionality and configuration of the different classes of events. You can save and print the log files.

For more information on the Log file, refer to the *EXPRESS Administrator's Guide*.

Trace file

The Trace file captures data traffic at either the link or station levels on the current configuration.

Data traffic information includes ACTPU, ACTLU, BIND, and UNBIND information. This file also provides the basic functionality and configuration of the different classes of events, and allows you to save and print the log files.

For more information on the Trace file, refer to the *EXPRESS Administrator's Guide*.

Background process debugging

Background process debugging allows you to view the information received at each background process. You activate this type of debugging by modifying the `express_adm` script. Specific instructions are included in the script itself.

Use this type of information carefully since it creates and outputs information to a boundless ASCII file. When you use this process, terminate the data collection as soon as you acquire the necessary information.

Traceable processes

Table 5-1 provides information on the processes that can be traced, the default output file, and the type of information generated.

Table 5-1: Traceable processes

Process	Default output file	Functionality
cdmserver	cdmserver.out	Records database read/write operation
executive	executive.out	Records event messages and requests for resources
agent	agent.out	Records log messages from kernel SNA communication startup through XID negotiations
watchdog	watchdog.out	Records 3270/EHLLAPI messages
jobex	jobex.out	Records RJE related operations
tn3270d	tn3270d.out	Records EXPRESS tn3270 Server messages

Debugging levels

There are different levels of debugging capability that each process supports. Table 5-2 lists the debugging levels and information available at the different levels.

Table 5-2: Debugging levels

Debug Level	Information Presented
0	No Debugging
1	Basic debugging
2	Inter-process messages
3	Messages from the Kernel
5	Reserved
7	Most verbose level of debugging; also provides the most information. Should only be used when requested by the vendor.

Host communication messages

This section contains a list of host communication error messages and the suggested actions you can take to correct problems encountered.

Condition	After installation, EXPRESS will not come up.
<i>Action/ Explanation</i>	If you did the installation as root, root is not an EXPRESS user. Either switch to user express (express -u express), or define root as an EXPRESS user.
Condition	Console Message: All communications boards are not operational.
<i>Action/ Explanation</i>	Usually, this indicates an IVR Generator reset performed without EXPRESS running. This host link has come up, but since EXPRESS is not running, there are no sessions. Start EXPRESS, then do an IVR Generator reset.
Condition	Console Message: The host connection was lost.
<i>Action/ Explanation</i>	If you encounter this when you start EXPRESS, it indicates that EXPRESS is waiting for you to dial in to the host (for a dial up SDLC connection). Otherwise, it indicates that the host connection was, in fact, lost.
Condition	Console Message: Unable to find ACTION (template name).
<i>Action/ Explanation</i>	The host link cannot find the action template. The host link will only notify you of a missing action template if it is referenced by <code>trs.conf</code> . Otherwise, a missing action template can go unreported.

Condition	Console Message: Unable to find SCREEN (template name).
Action/ Explanation	The host link cannot find the screen template (template name) .scn. The host link <i>always</i> reports a missing screen template because it checks all action templates for screen templates, then verifies their presence.
Condition	At start-up time, trs.log reports that no sessions are active.
Action/ Explanation	The vendor communication initialization failed. First stop IVR Generator, restart the vendor software, and then restart IVR Generator. For information on starting and stopping IVR Generator, refer to the <i>System Administration Guide</i> . For information on starting and stopping EXPRESS/SSI, refer to the <i>EXPRESS Administrator's Guide</i> .
Condition	Extended errors appear in trs.log after start-up.
Action/ Explanation	These messages are sent from the vendor software to TRS to signal an error occurring at the lower levels. Create and gather a vendor log file (see vendor documentation for specifics).
Condition	A logical unit that has been successfully logged on is now consistently reporting a status of LOCKED.
Action/ Explanation	Usually, when a LU locks, an application cannot verify all possible error cases and continues to function as if no error occurred. By continuing to send data, the host buffer fills, causing the LU to lock. Verify that your application screen templates check for a unique key on each screen. Also, verify that the host responded to the ping action.

Condition	The message “HOST connection is lost” appears in the trs.log.
<i>Action/ Explanation</i>	A lost connection with the host computer occurred. This is an acceptable message. Wait until the host computer is active again. TRS should reattach itself. Typically, it executes the logout action template, since TRS receives the login screen when reattachment occurs.
Condition	The host link enters logout loop after reconnection.
<i>Action/ Explanation</i>	Screen(s) generated by the host was unexpected by the application. Stop and then restart all components (IVR Generator, TRS and vendor software). Be sure to include this unexpected screen in your logout action template for the next time.
Condition	Scripts that auto start EXPRESS/SSI sessions do not execute properly or only execute by using the sh command as a prefix.
<i>Action/ Explanation</i>	The execute bit is not set on the file. Use chmod to set the executable bit.

Condition	EXPRESS/SSI background processes are growing.
Action/ Explanation	Use the <code>ps -aux</code> command to determine this. A memory leak occurred in these EXPRESS/SSI background processes. EXPRESS/SSI release 2.04.b has this problem. The EXPRESS/SSI software needs to be upgraded to version 2.04.c or greater to resolve this issue. To determine the version of EXPRESS/SSI software executing on the applications processor, do the following: Use the <code>scn <process></code> command where process is an EXPRESS/SSI background process (agent, watchdog). The command <code>scn agent</code> produces a result that looks similar to Release xx SCN <version> agent. EXPRESS/SSI version 2.04.c has an SCN number of 2965. Anything less than that number is 2.04.b and should be updated.
Condition	Host reconnection is not occurring.
Action/ Explanation	The vendor software was not configured to keep retrying reattachment. Make sure the vendor software is configured to keep trying reattachment. Use one of the provided scripts to force the vendor software to attempt reattachment.

Condition **When more than 64 logical units are defined, an error message appears indicating all EXPRESS/SSI users greater then 64 cannot start.**

*Action/
Explanation* The following are configuration parameters in EXPRESS/SSI software:

Under

Users/Sessions

Terminal Emulation Services

Standard

There is a limits option. Modify that field so that all users can be started successfully.

Appendix A: List of terms

This section lists terms and definitions related to the information presented in this guide.

3270 Protocol

A set of communication rules used by workstations designed for use with the IBM mainframe computers.

5250 Protocol

A set of communication rules used by terminals designed specifically for use with application system computers (for example, AS/400), as well as earlier IBM system computers (36 and 38).

ACTLU

SNA command code used to activate the logical unit.

ACTPU

SNA command code used to activate the physical unit.

AID

Attention identifier; the key that interrupts the operation of the host to process the current terminal input.

application

With respect to IVR Generator, an application is a program that controls the activity on one or more telephone trunks connected to an MRS. With respect to a host computer, it is any type of program that carries out a task.

Application Programming Interface (API)

Software libraries that enable the development of applications in a new environment, which communicate with applications in the legacy system.

Application server

A computer or workstation running IVR Generator.

bind

SNA command code used to initiate a session.

branch

A pathway between cells in an IVR Generator application.

caller

A person whose phone call is received or originated by an IVR Generator application.

cell

The basic element of an IVR Generator application. Each cell performs an action, such as playing a prompt to a caller. Each cell has a set of branches to other cells. After the cell performs its action, it determines which branch the application should follow to the next cell.

channel

A telephone trunk within a cluster of MRSs.

COMA cell

IVR Generator cell that aborts a transaction with an SNA host application.

COMI cell

IVR Generator cell that sends input to an SNA host application through the TRS process.

COMO cell

IVR Generator cell that receives output from an SNA host application through the TRS process.

EHLLAPI

Extended High Level Language Application Programming Interface allows you to search through the host screens for specific data and conduct actions on the data. This is also called screen scraping.

emulation

Imitating a computer or computer system with a combination of hardware and software. Emulation allows programs written for one computer to be run on another.

host

A networked computer that provides applications and services to other networked computers. In this guide, a host is an IBM computer that supports the SNA communication protocol.

link

A physical connection between two devices.

LU

Typically, an SNA terminal or print device.

LU6.2-based file transfer

A protocol for high-performance cooperative processing, specifically as a file transfer mechanism. The cornerstone of LU6.2 is the concept of a pair of transaction programs (TCP/IP and SNA); file transfer and print sharing are implemented with companion programs, one on each target system.

MRS

Multimedia Resource Server.

node

A grouping composed of an application processor connected to one or more MRSs.

screen scraping

See EHLLAPI.

sense code

SNA error code returned by the IBM host application.

session

A connection between two systems for the exchange of user data. Sessions are between logical units (LUs).

A connection to a host as defined in the `trs.conf` file, representing a terminal connection.

SNA

Systems Network Architecture. A group of IBM protocols that define how IBM computers (and any other computer that uses the SNA protocol) transmit and receive data. This protocol is also used by many other computer hardware and software manufacturers. It presumes an underlying transport such as SDLC, Token Ring, Ethernet, or X.25.

station

The physical device to which you communicate.

TCP/IP (Transmission Control Protocol/Internet Protocol)

Originally developed for U.S. military communication, this protocol has become a defacto standard worldwide and the protocol of choice for backbone routing of data traffic. It presumes an underlying communications capability such as Token Ring or Ethernet.

templates

ASCII files used by the TRS process to manage the SNA terminal session.

terminal emulation

Allows users in a non-IBM environment to log in the applications running in the legacy (IBM) environment just as if they had legacy environment terminals (3270 and 5250).

transaction

The function performed by a set of action and screen template files when executed by TRS.

TRS

Terminal Resource Server. IVR Generator process that manages communication between an IBM host and the IVR Generator Application Processor.

unbind

SNA command code used to terminate a session.

Index

Symbols

#comment, 3-11, 3-27

A

Action templates, 3-2, 3-3, 3-4, 3-7, 3-10

 Action, 3-5, 3-16

 checking with trs -c, 3-36

 creating, 3-6

 Heartbeat, 2-15, 3-6

 Initial-Action, 3-5, 3-7, 3-14

 sample, 3-14

 Keep-Alive, 3-6

 Logout-Action, 3-5, 3-18

 Reset-Action, 3-5, 3-17

 sample, 3-18

 syntax, 3-11

 types of, 3-5

application

 development

 IVR Generator, 1-3

 voice, 3-2

 server, 1-2

 synchronous SNA protocol, 1-2

 TRS, 1-2

 TRSC, 1-2

architecture

 flow, 1-4

C

COM cells

 COM

 reply codes, 4-9

 COMA, 1-5, 4-8

 parameters, 4-8

 using, 4-8

 COMI, 1-5, 3-29

 parameters, 4-2

 reply codes, 4-9

 setting, 4-2

 COMO, 1-5, 3-29

 cell branches, 4-7

 parameters, 4-5

 reply codes, 4-9

 setting, 4-5

 sample application, 4-10

 testing communication, 4-9

 using, 4-1

communications

 adapter, 1-7, 2-3

 media, 2-2

D

debugging

 levels, 5-3

 see troubleshooting, 5-3

E

EXPRESS

 checking links and stations, 2-13

 components, 2-4

 configuring, 2-8

 defining users and sessions, 2-12

 installation, 2-3

 naming convention, 2-8

- software, 1-4
- starting, 2-7
- tools, 5-1

F

- field, 3-22
 - writable, 3-22
- field-descriptor syntax, 3-29
- files
 - log, 5-1
 - lubad.dat, 2-18
 - lubaf.dat, 2-18
 - template, Action, 3-1
 - trace, 5-1
 - trs.conf, 2-14, 3-4
 - syntax, 2-14
 - trs.conf example, 2-15
 - trs.sdf, 2-16
 - checking with trs -c, 3-36
 - sample, 2-17
 - setting up, 2-16
 - verbose mode, 5-1
- function keys, 3-32

H

- hardware, 1-2
 - defining, 2-8
 - installation, 2-3
 - specifications, 1-2
- HELLAPI, 1-7
- host communication
 - application, 1-1
 - configuring IVR Generator, 2-14
 - links, 1-1
 - starting session, 2-14
 - verifying connections, 2-13
- host information, 2-2
- host link, 1-6
 - architecture, 1-4
 - communications, 1-1, 2-1
 - configuring, 2-1
 - installing, 2-1

I

- IBM
 - application, 1-6
 - mainframe, 1-3
 - menu-driven program, 1-3
 - protocol, 1-7
 - screens, 1-6
 - system administrator, 1-4
- ID
 - see login, 2-18
- Initial-Action template
 - see Action templates, 3-5, 3-7, 3-14
- IVR Generator
 - applications, 1-7
 - call flow, 1-3, 1-4, 1-6, 4-10
 - COM cells
 - see COM cells, 1-4
 - see application development, 1-3
 - starting, 2-22

L

- log file
 - see files, 5-1
- login
 - ID, 2-18
 - password, 2-18
- Logout-Action template
 - see Action templates, 3-19
- LU 6.2
 - defining, 2-10
- lubaf.dat file
 - see files, 2-18
- lubuf.dat file
 - see files, 2-18

M

- messages
 - host communication, 5-5
- mode
 - manual, 3-13
 - verbose, 5-1

P

password
 see login, 2-18

R

Reset-Action template
 see Action templates, 3-5, 3-17

S

SAM, 4-9
screen scraping, 1-7
Screen templates, 3-3, 3-4, 3-13, 3-22
 checking with trs -c, 3-36
 field-descriptors, 3-29
 Home Screen, 3-17
 moving around, 3-22
 row, column, 3-22
 sample, 3-32
 syntax, 3-27
 transparent functions, 1-7
SDLC, 2-3
sessions, 2-15
site planning, 2-2
 requirements, 2-2
SNA, 1-1
 defining a links and stations, 2-10
 defining a node, 2-9
 protocols, 3-4
 terminals, 1-3
Software
 EXPRESS, 1-6
specifications
 see hardware, 1-2
start-up script, 2-19
syntax
 see Action templates and Screen templates,
 3-27
System Activity Monitor
 see SAM, 4-9

T

templates
 developing, 1-5
 files, 3-1
 see Action templates, 3-11
 see Action templates and Screen templates,
 3-2, 3-3, 3-4, 3-7
 see Screen, 3-4
 see Screen templates, 3-3, 3-13, 3-22
terminal
 emulation, 1-7
 setting limits, 2-12
Token Ring, 2-3
trace file
 see files, 5-1
traceable processes, 5-3
transactions, 1-5, 3-17
 determining, 3-1
 sample, 3-2
 sample steps, 4-10
troubleshooting, 5-1
TRS, 1-2, 1-6
 modules, 1-4
 start-up switches, 2-22
 template file, 3-1
trs, 2-12
TRS log, 3-36
trs.conf file
 see files, 2-14, 3-4
trs.sdf file
 see files, 2-16
TRSC, 1-2, 1-6

V
validation tag, 3-23
 offset, 3-28

W
writable field, 3-22

Nortel IVR 2.5/S

SNA Host Communications User Guide

Toronto Information Products
Nortel
522 University Avenue, 14th Floor
Toronto, Ontario, Canada
M5G 1W7

© 1997 Northern Telecom
All rights reserved

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

IVR and Nortel are registered trademarks of Northern Telecom in the U.S. and Canada. IVR Generator 2.0/S and Meridian 1 are registered trademarks of Northern Telecom. DEC and VT 420 are registered trademarks of Digital Equipment Corporation. HP, LaserJet, and ThinkJet are registered trademarks of Hewlett-Packard Company. X Window System and X are trademarks of the Massachusetts Institute of Technology. Motif is a trademark of Open Software Foundation Inc. UNIX is a registered trademark of AT&T.

Product release: 2.5/S
Document release: Standard 1.0
Date: July 1997

Printed in the United States of America

NORTEL
NORTHERN TELECOM